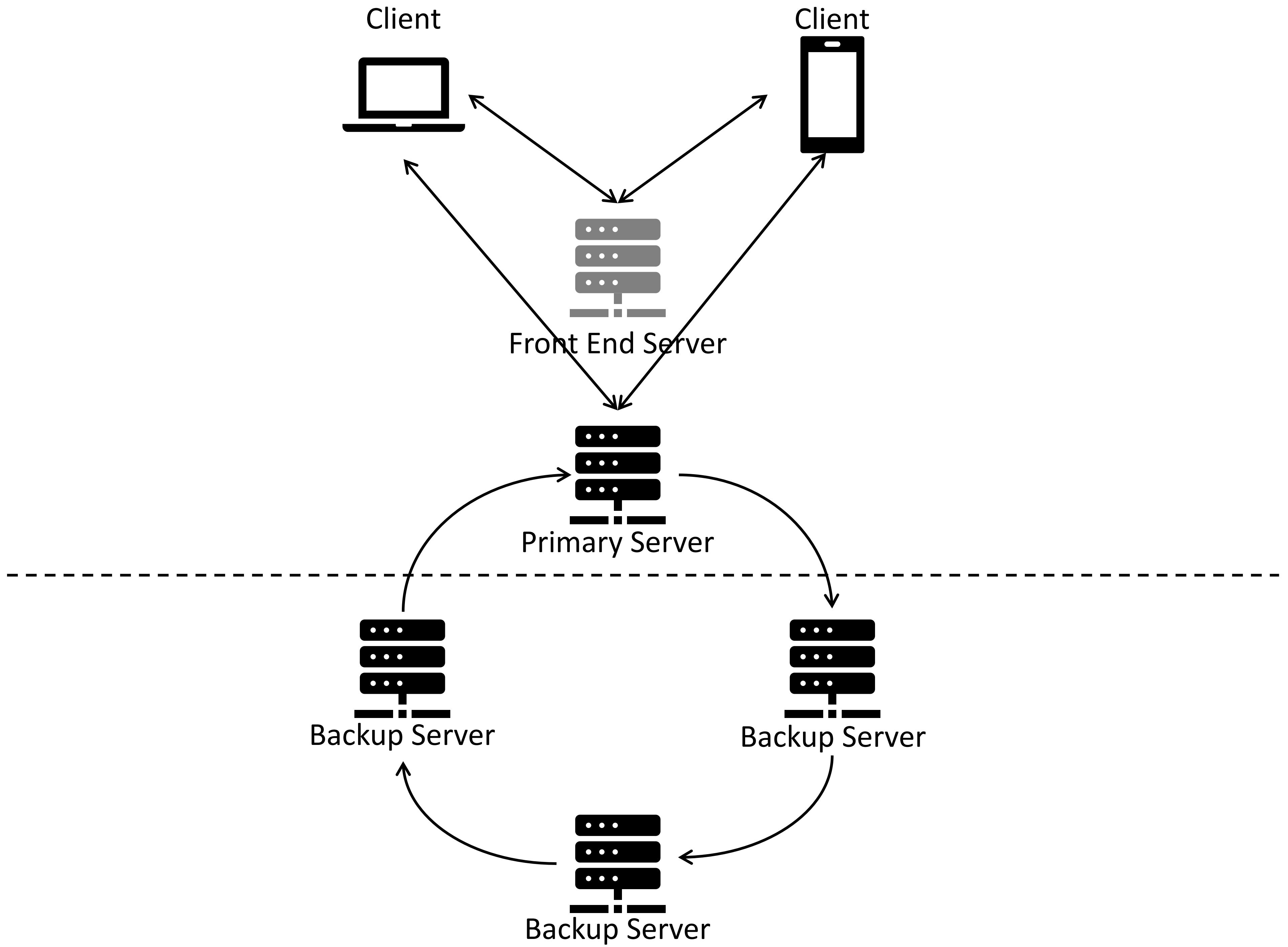
architectural viewpoints on

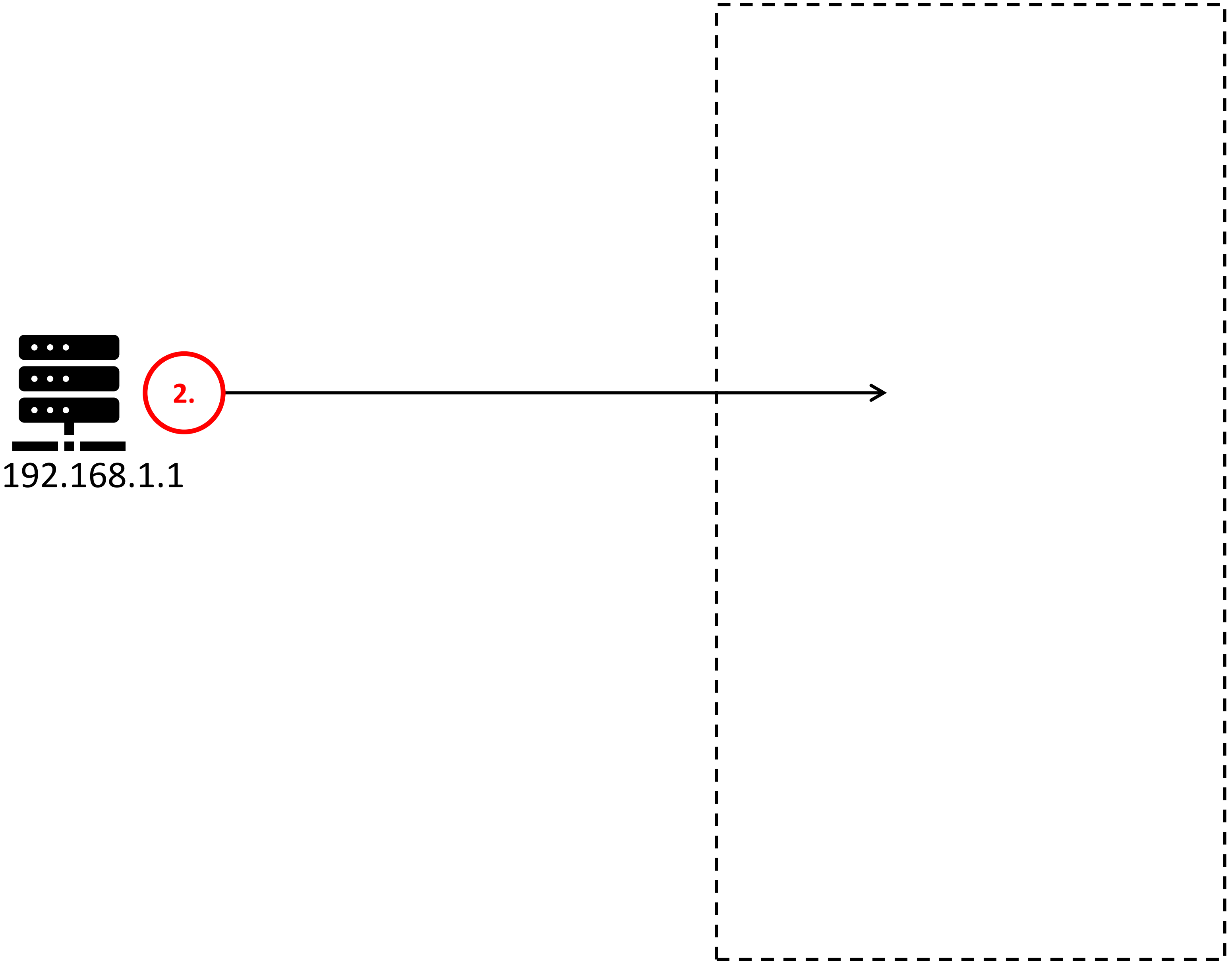
LOCALS-HIT

Architecture


Client
Client
Front End Server
Primary Server
Backup Server
Backup Server
Backup Server

Dynamic discovery

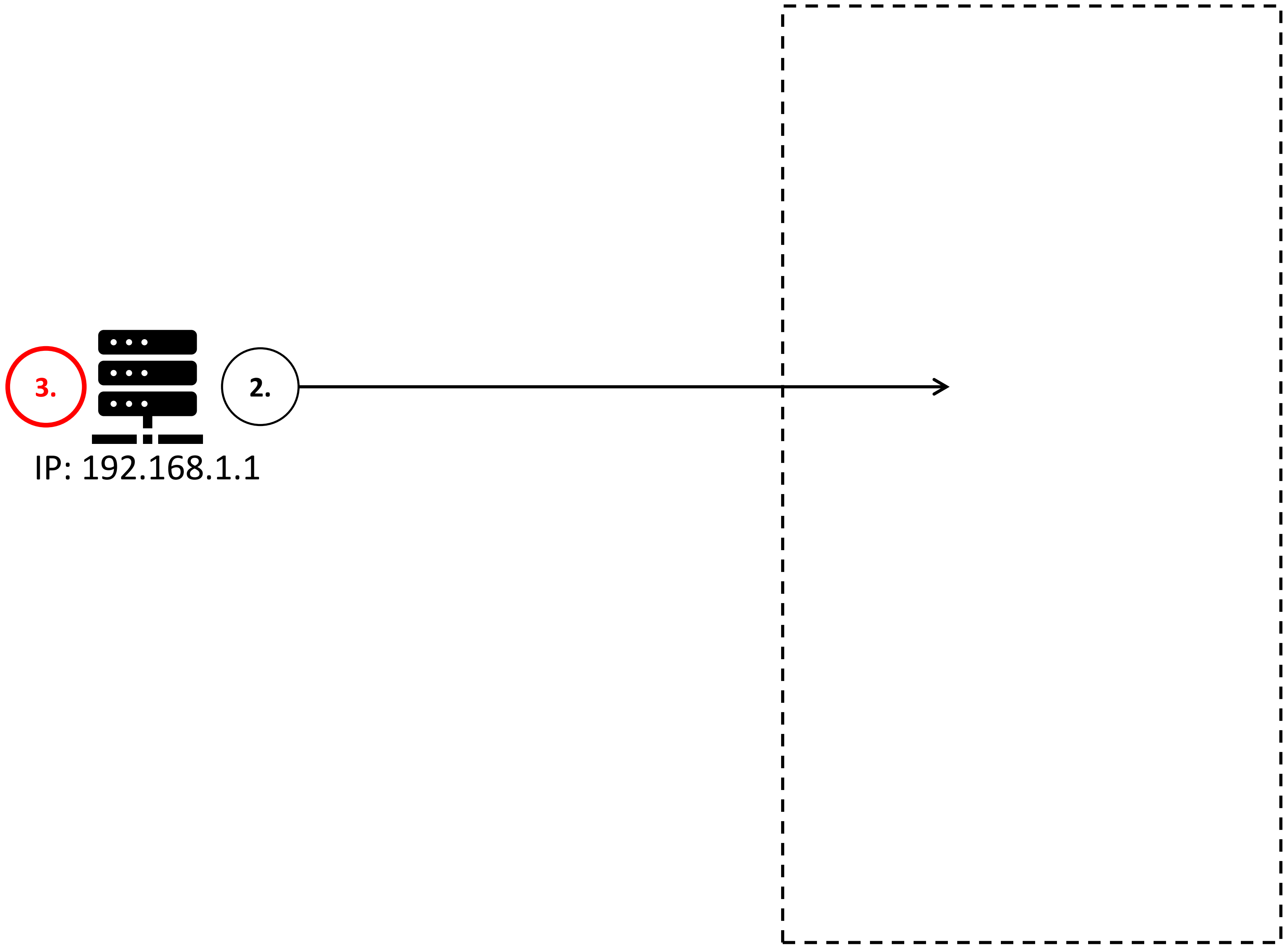
Dynamic discovery of hosts



2.

New participant sends a service announcement message SA(pid): SA(192.168.1.1)

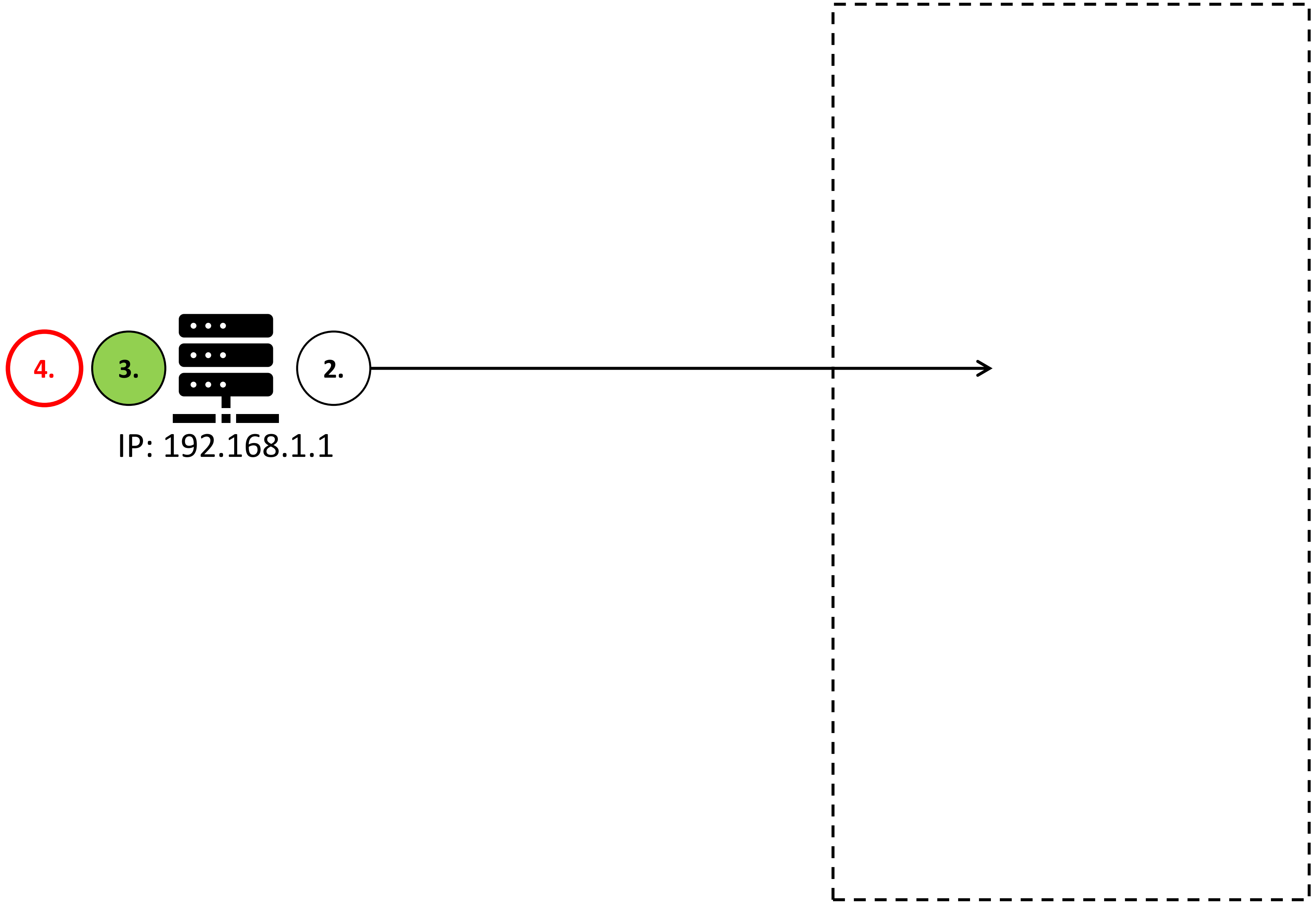
Dynamic discovery of hosts



3.

No response in time interval of two seconds → Server marks itself as the first and only participant in the system

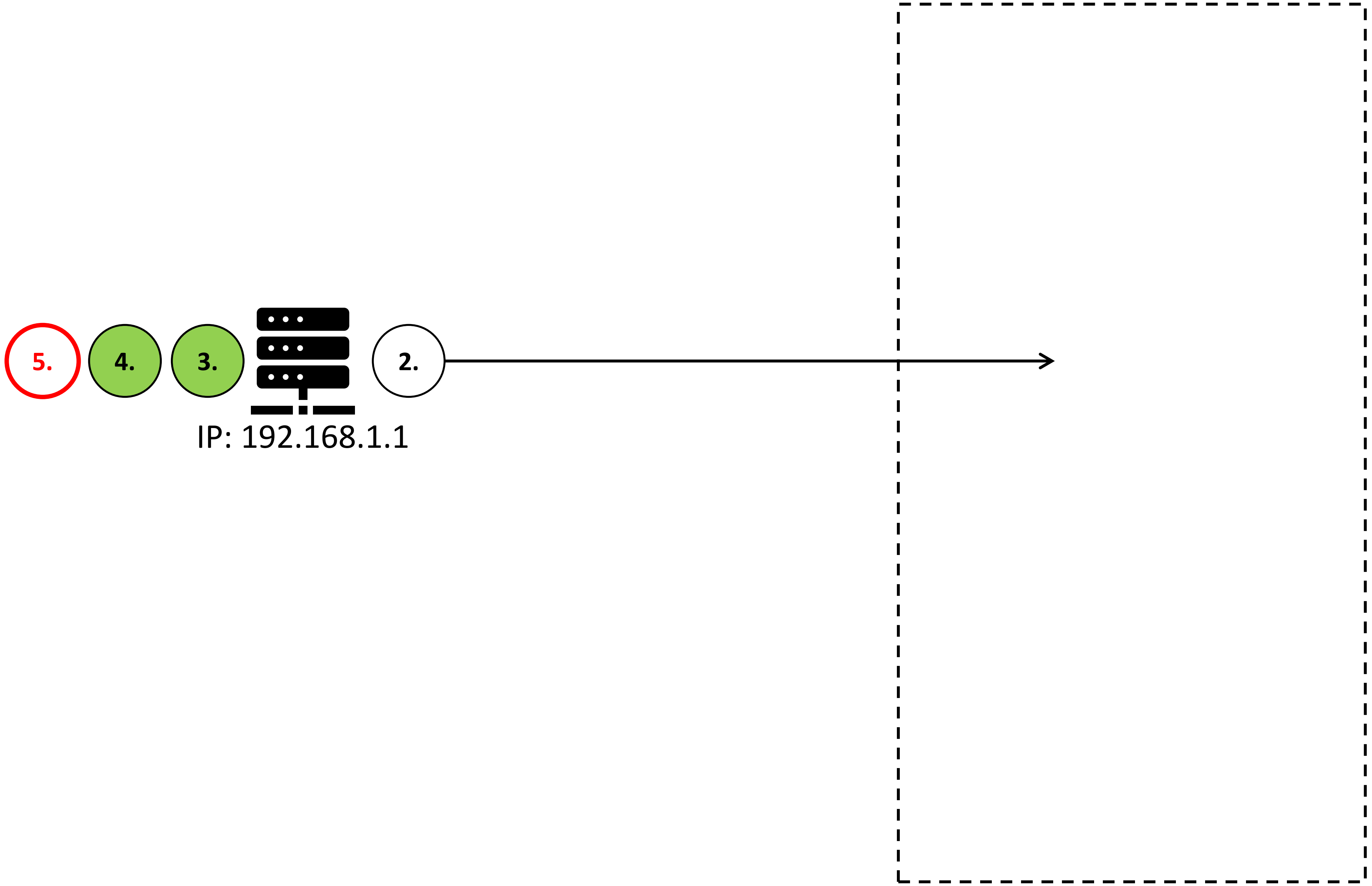
Dynamic discovery of hosts



4.

Server starts the ring formation and the leader election with itself

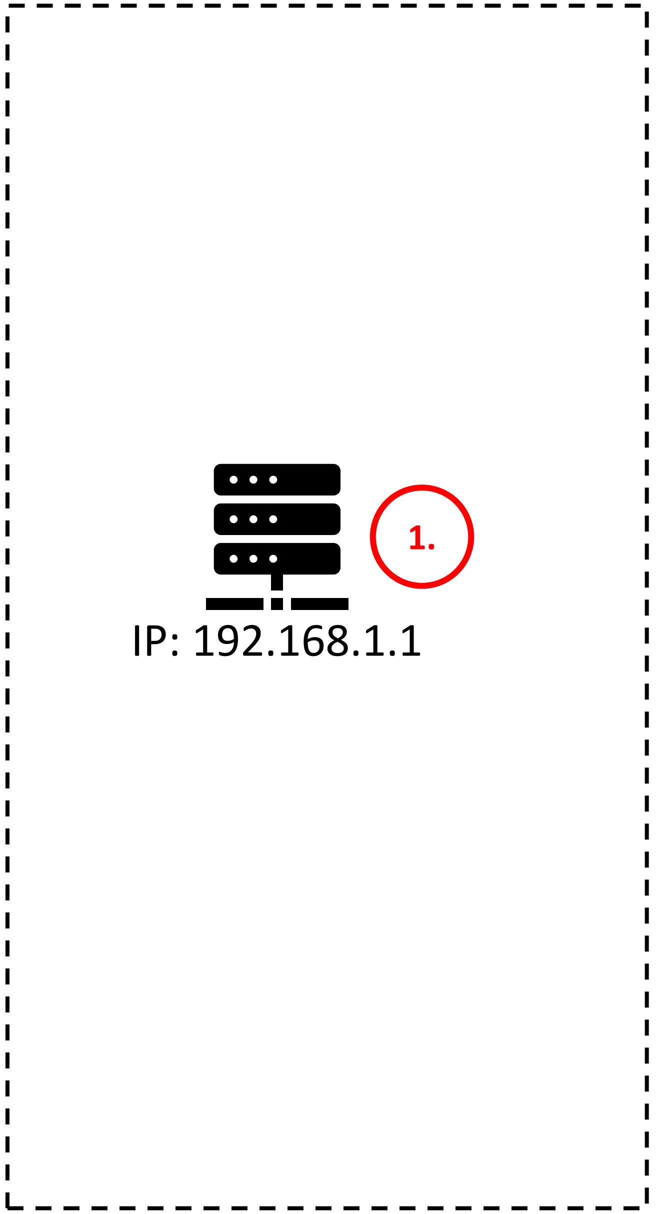
Dynamic discovery of hosts



5.

Server continuously listen for service announcement messages

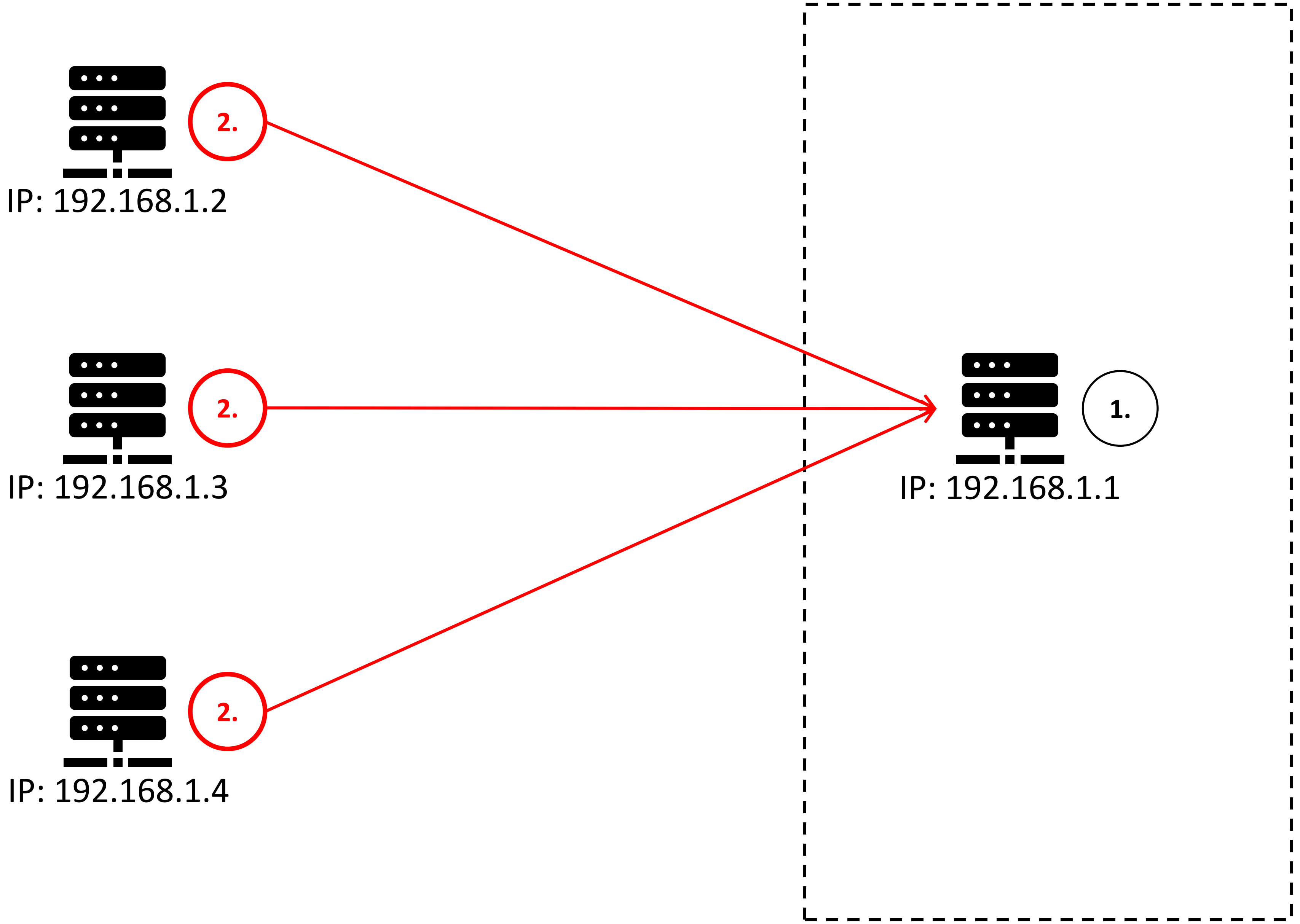
Dynamic discovery of hosts



1.

Server continuously listen for service announcement messages

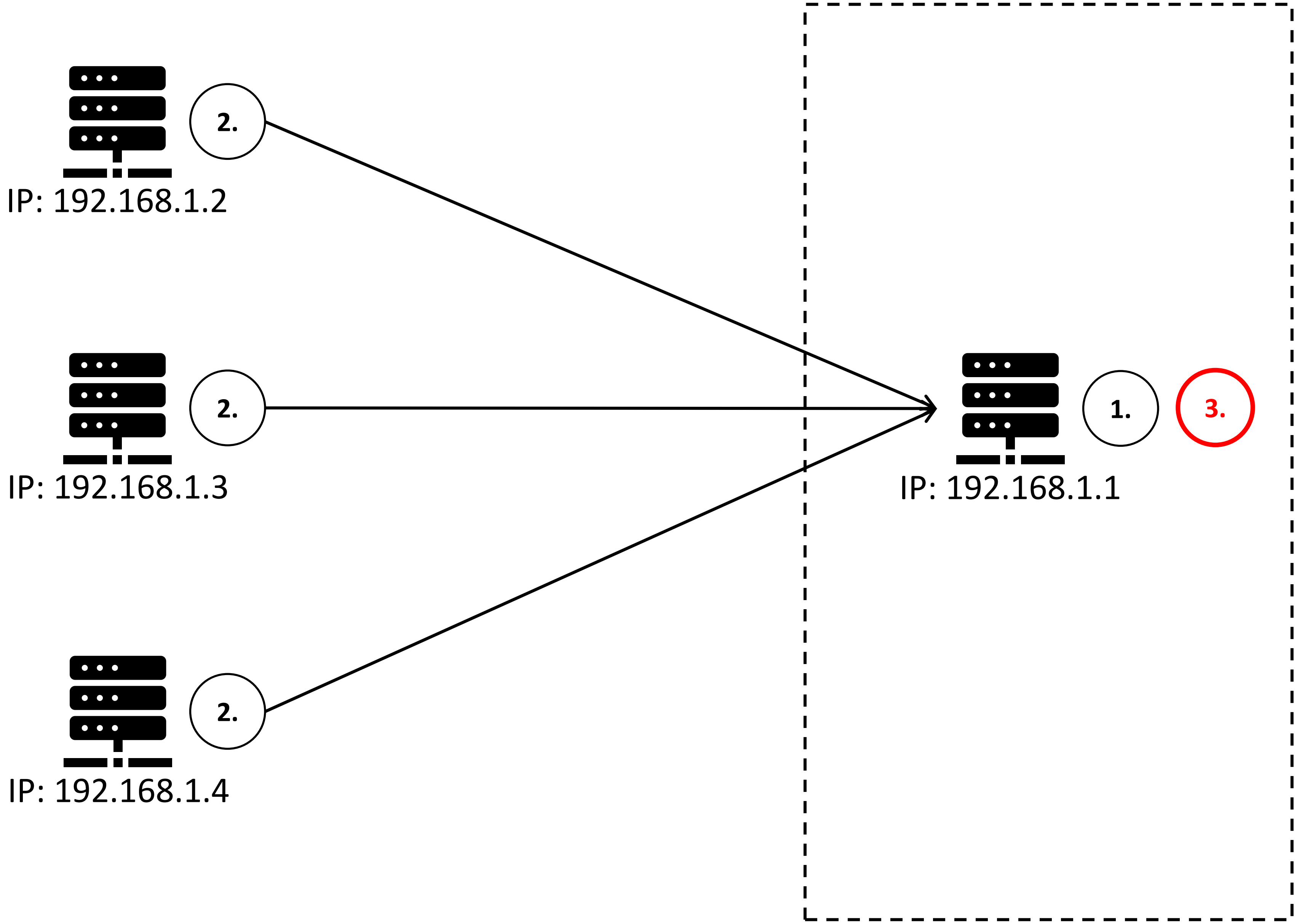
Dynamic discovery of hosts



2.

Each new participant sends a service announcement SA(pid):
SA(192.168.1.2)
SA(192.168.1.3)
SA(192.168.1.4)

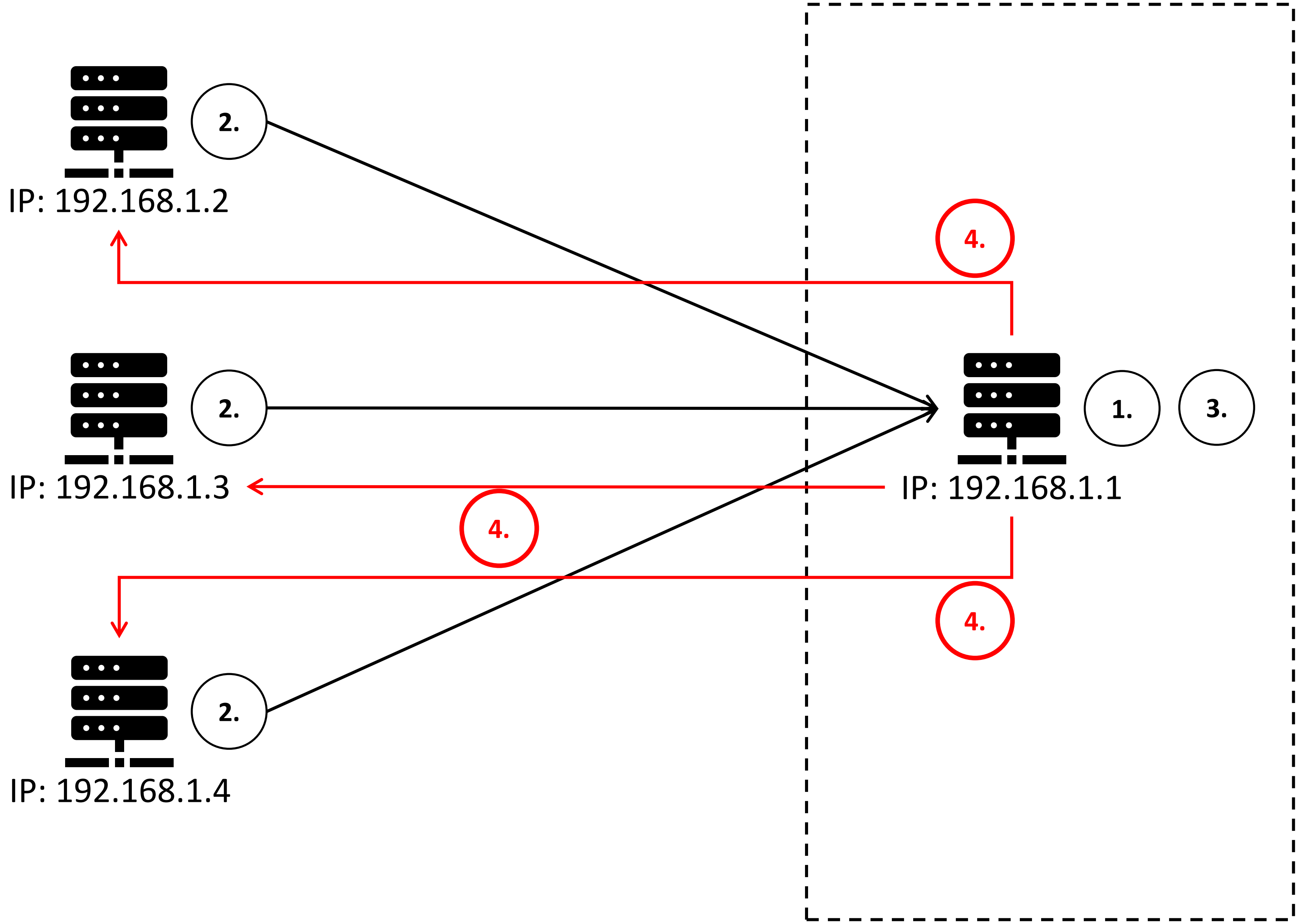
Dynamic discovery of hosts



3.

Each Recipient updates its group view:
192.168.1.1
192.168.1.2
192.168.1.3
192.168.1.4

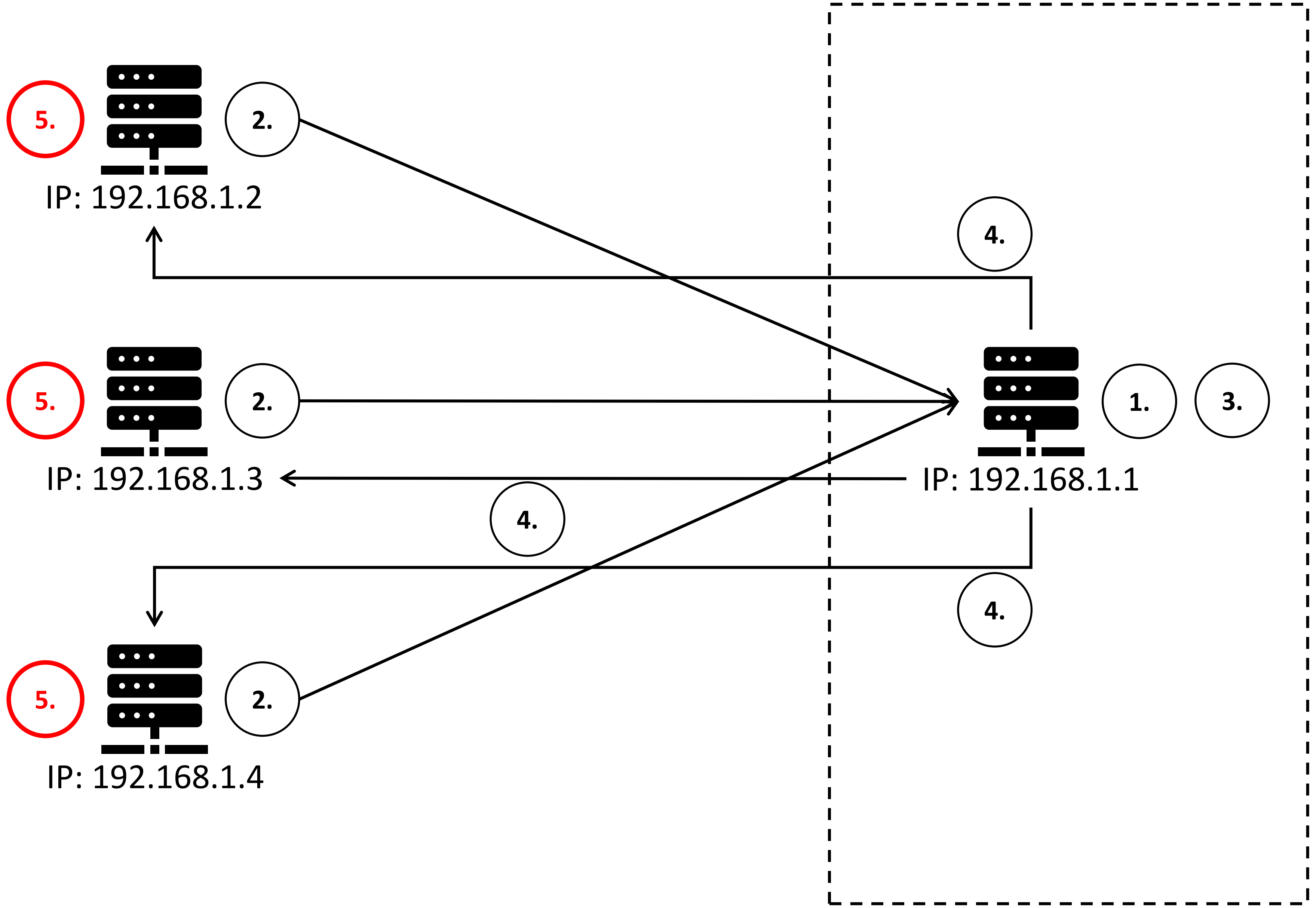
Dynamic discovery of hosts



4.

Each recipient sends a reply message RP(pid): RP(192.168.1.1)

Dynamic discovery of hosts

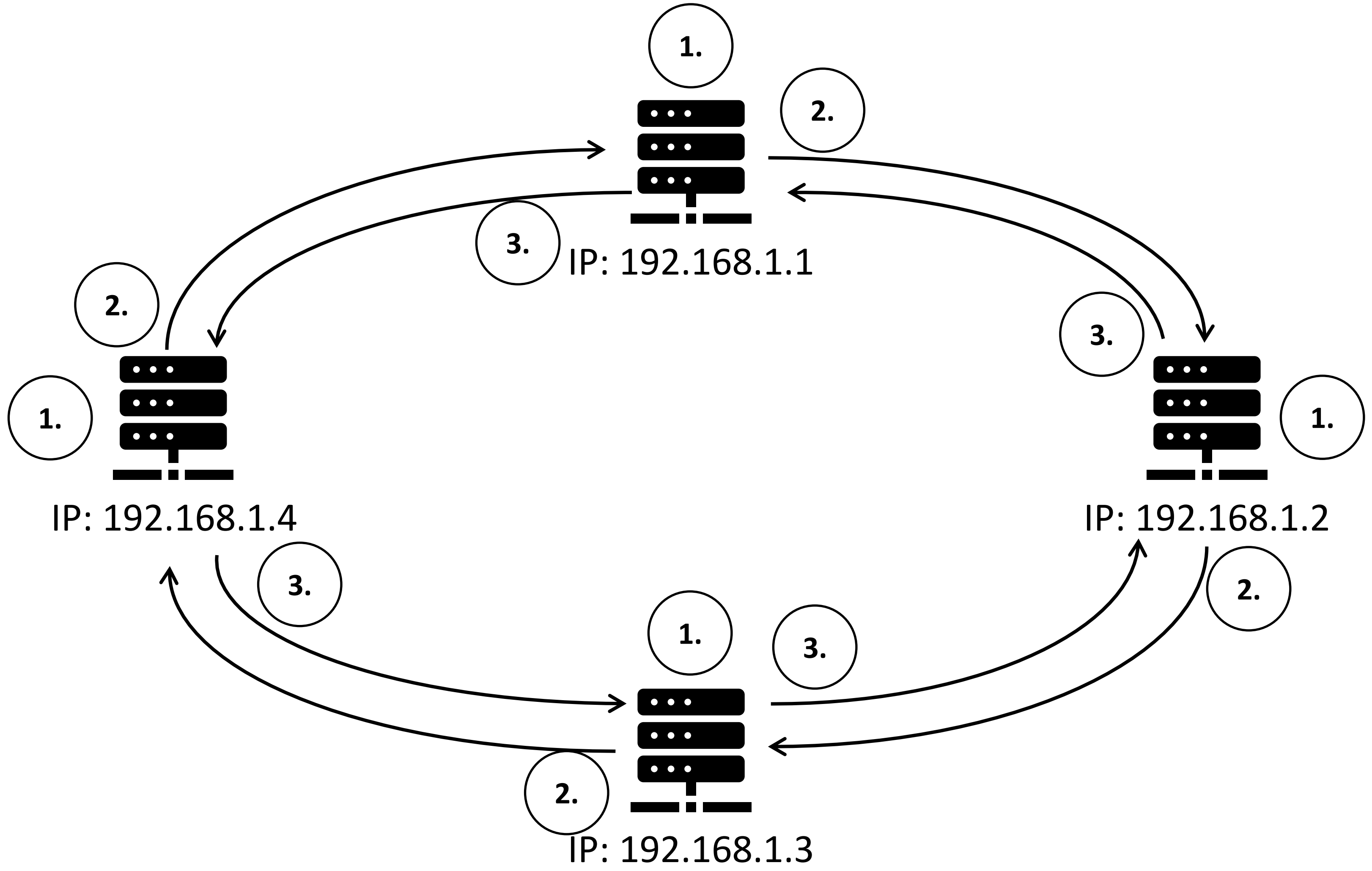


5.

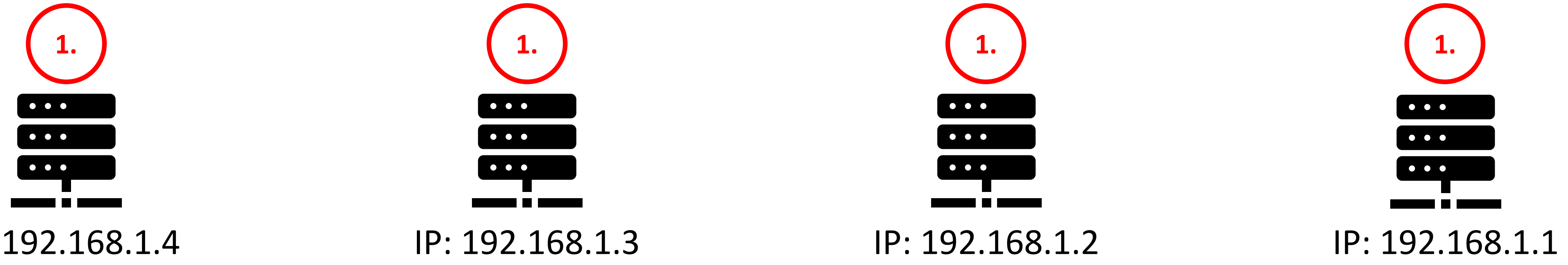
Each new Participant updates its group view:
192.168.1.1
192.168.1.2
192.168.1.3
192.168.1.4

Ring Formation

Ring formation



Ring formation



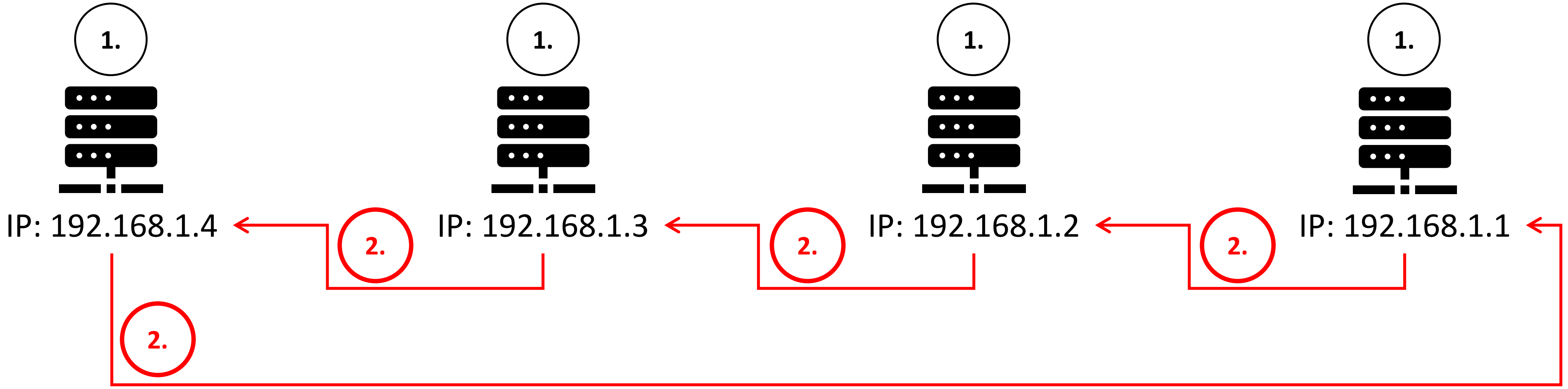
1.

Each Participant has the same group view:
192.168.1.1
192.168.1.2
192.168.1.3
192.168.1.4

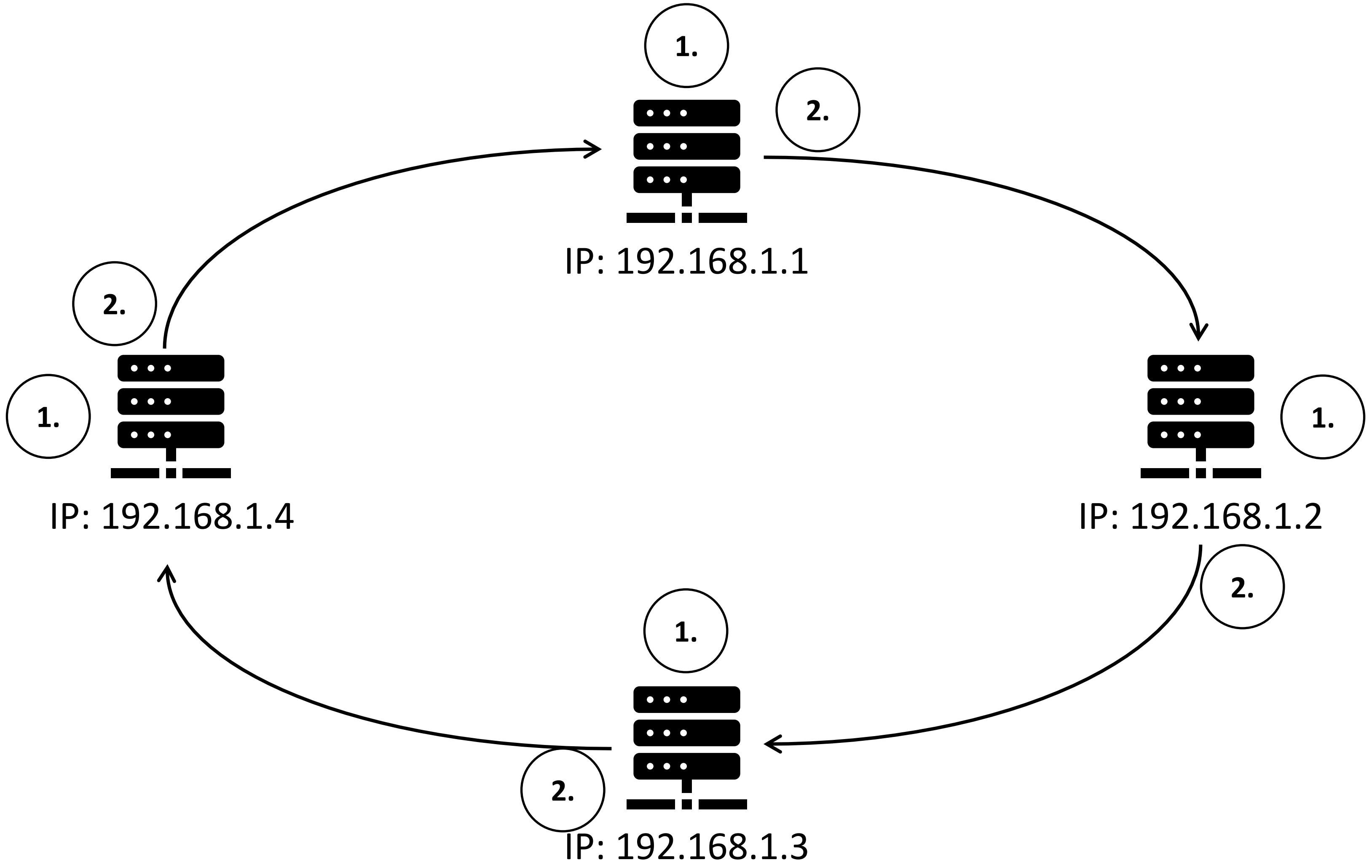
Ring formation

2.

The participants sort themselves in ascending order of their IP addresses → Each Participant knows its clockwise neighbour



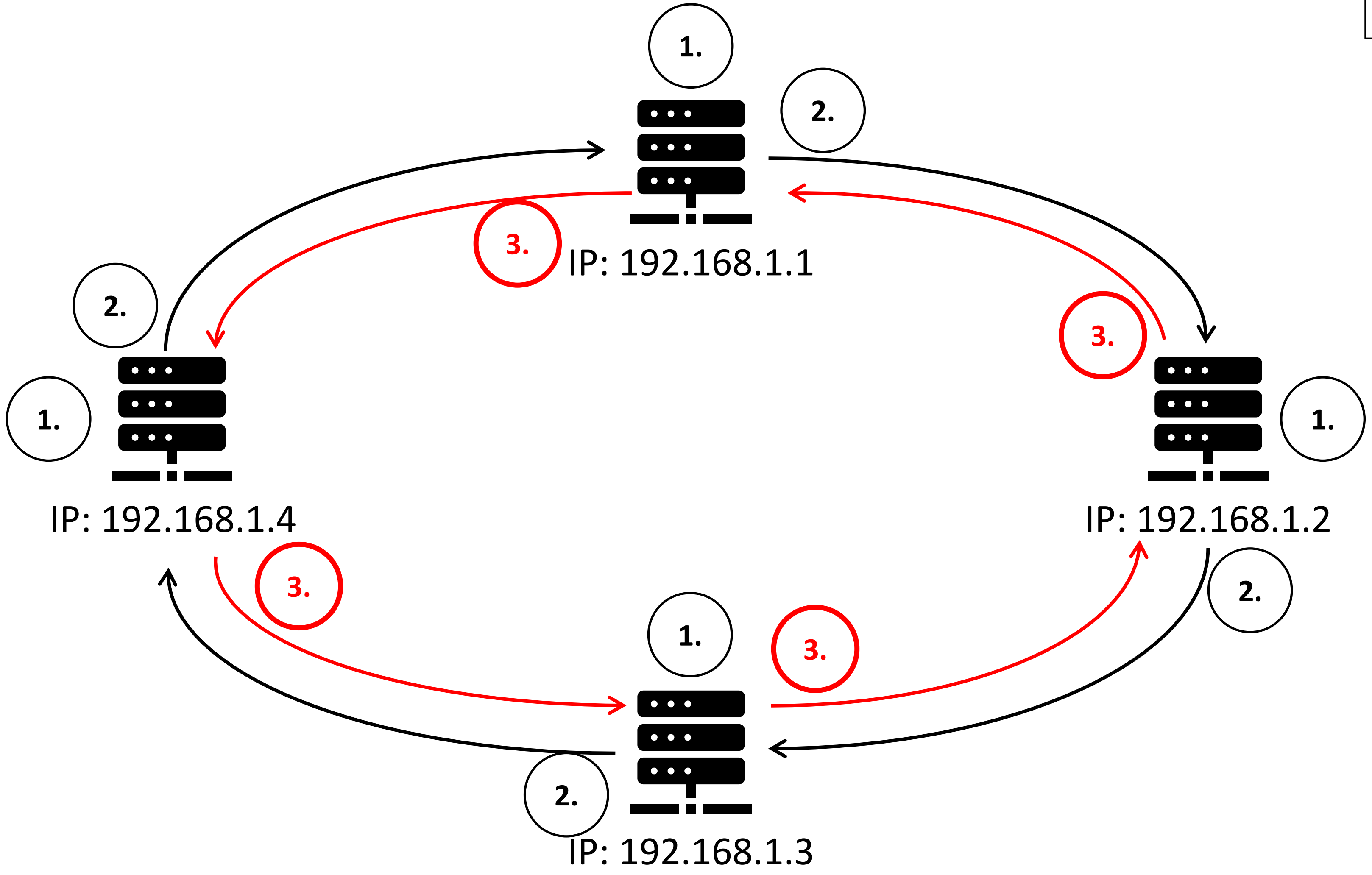
Ring formation



Ring formation

3.

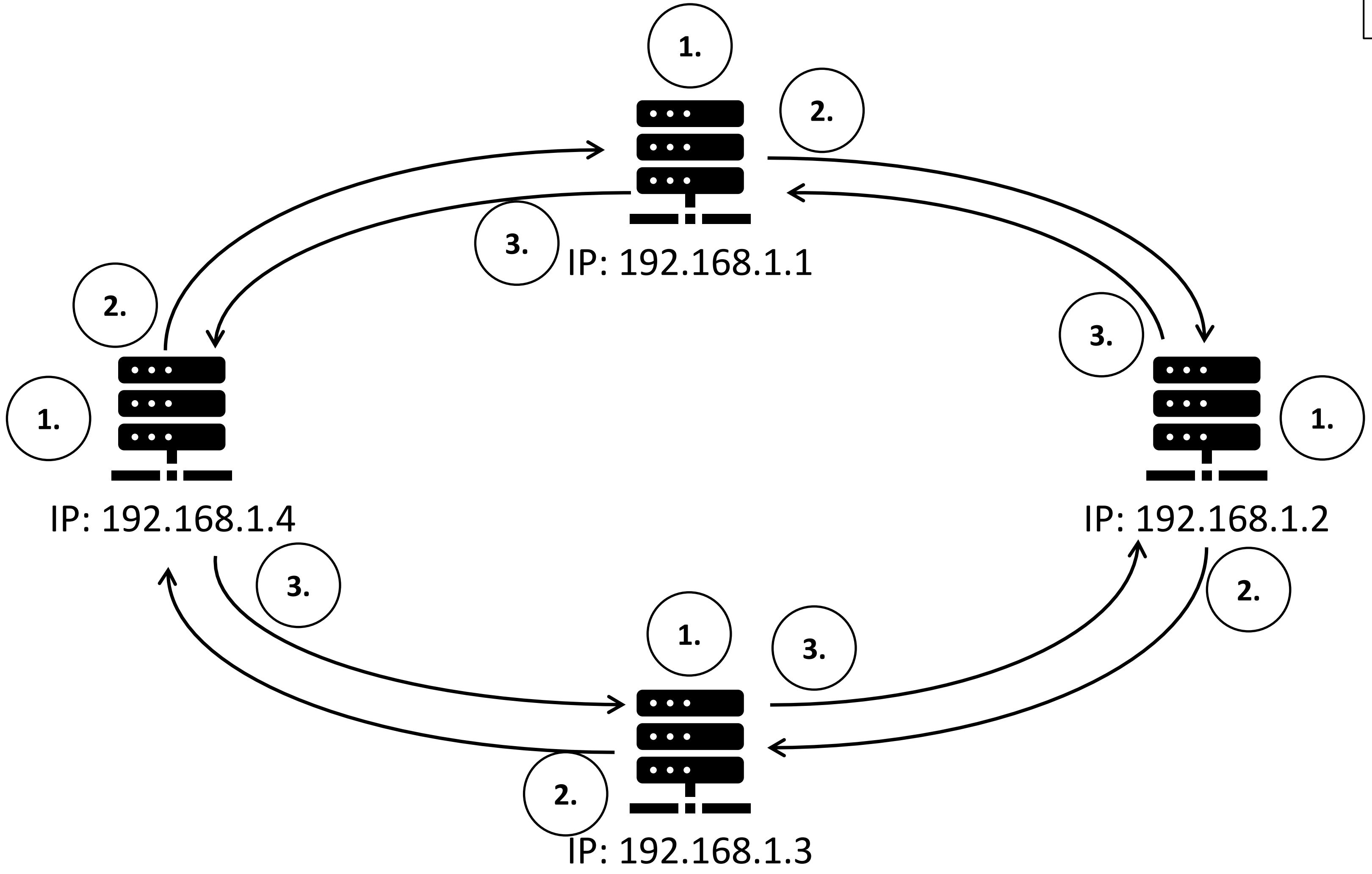
Each Participant knows its counter-clockwise neighbour



Ring formation

5.

Each participants records its clockwise neighbour by the received message

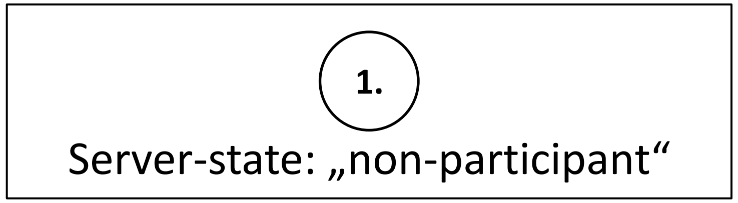


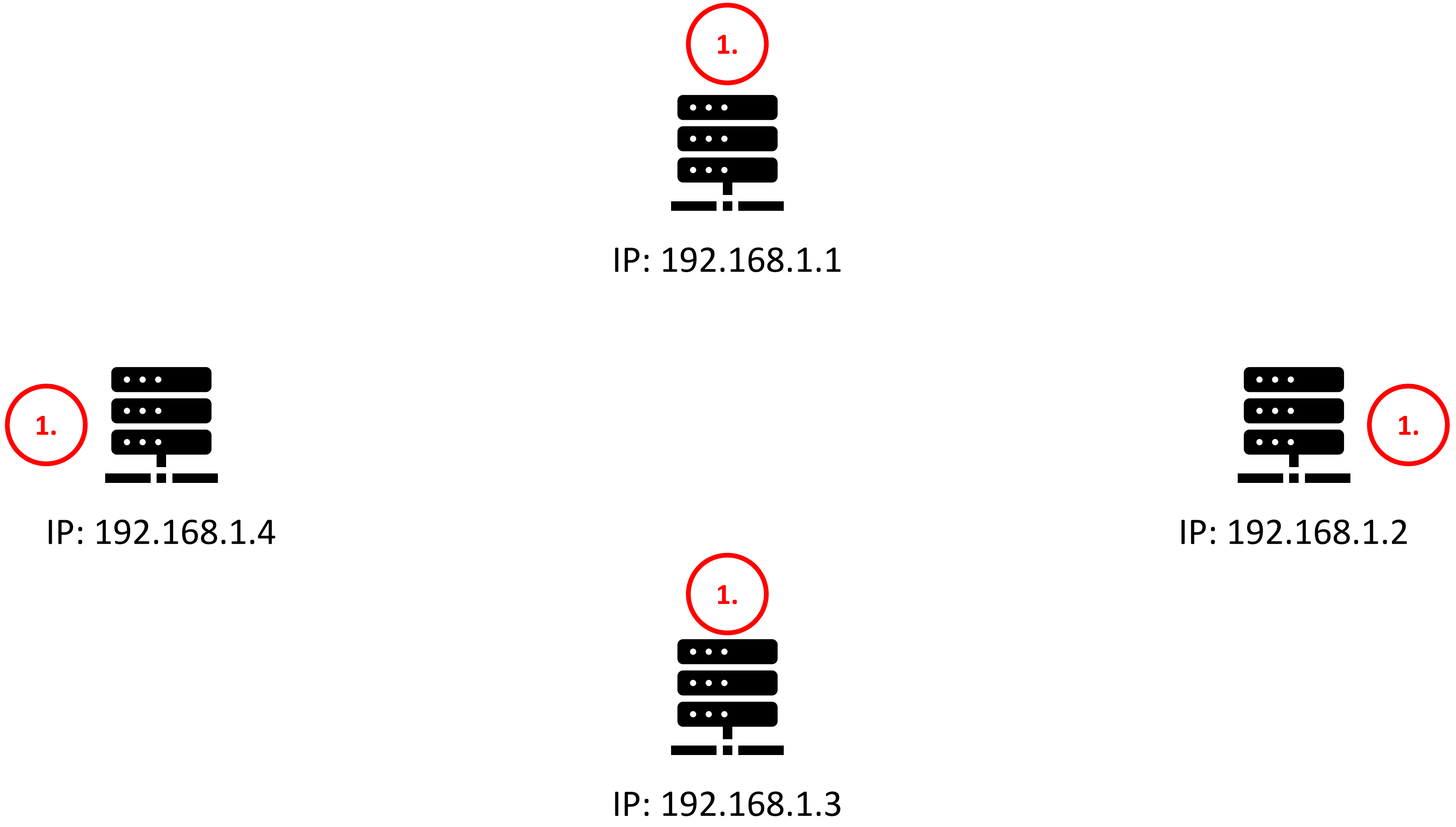
Leader Election: Chang and Roberts algorithm

„just one server starts an election“

Chang and Roberts algorithm

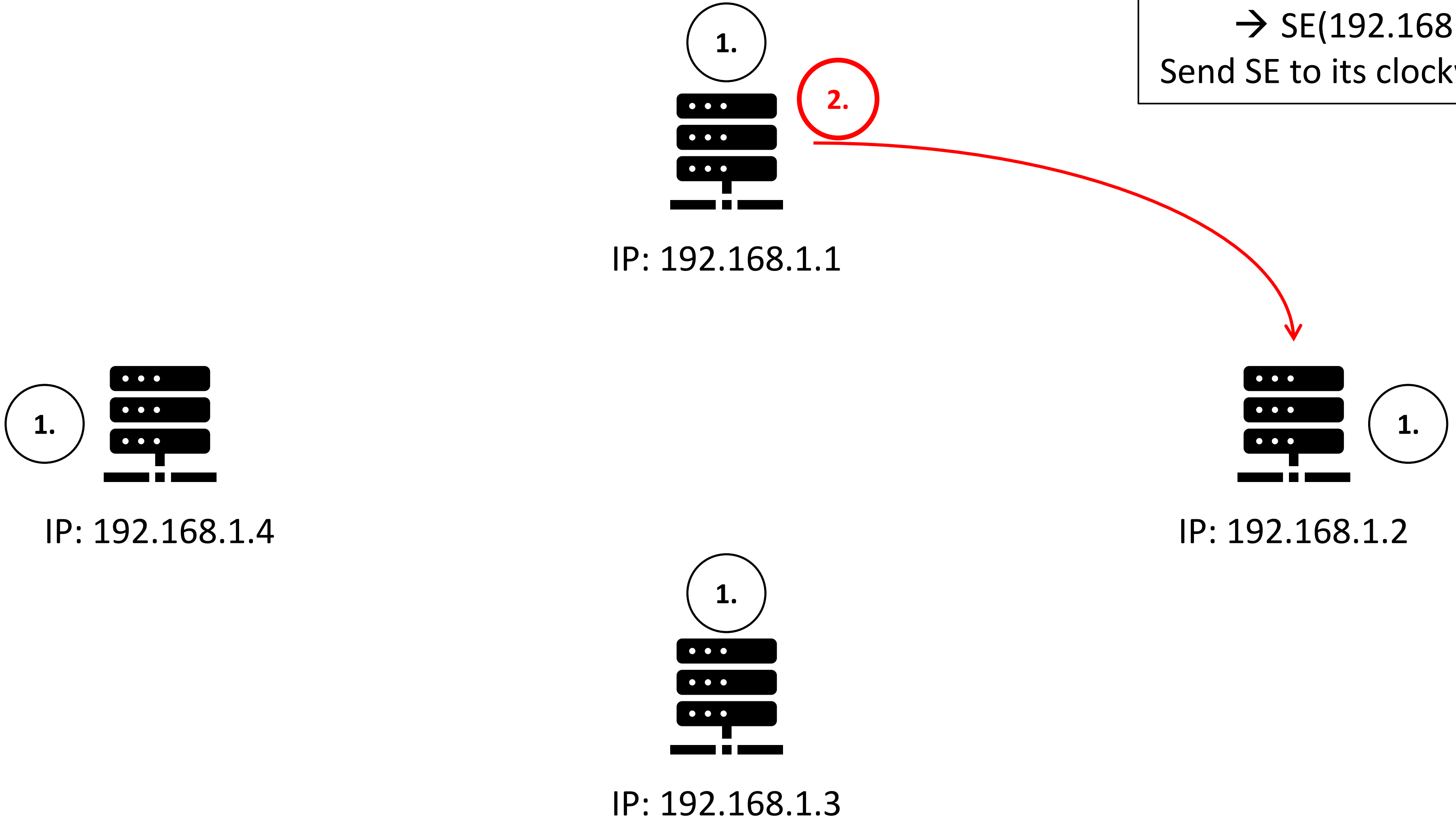
„just one server starts an election“





Chang and Roberts algorithm

„just one server starts an election“



2.

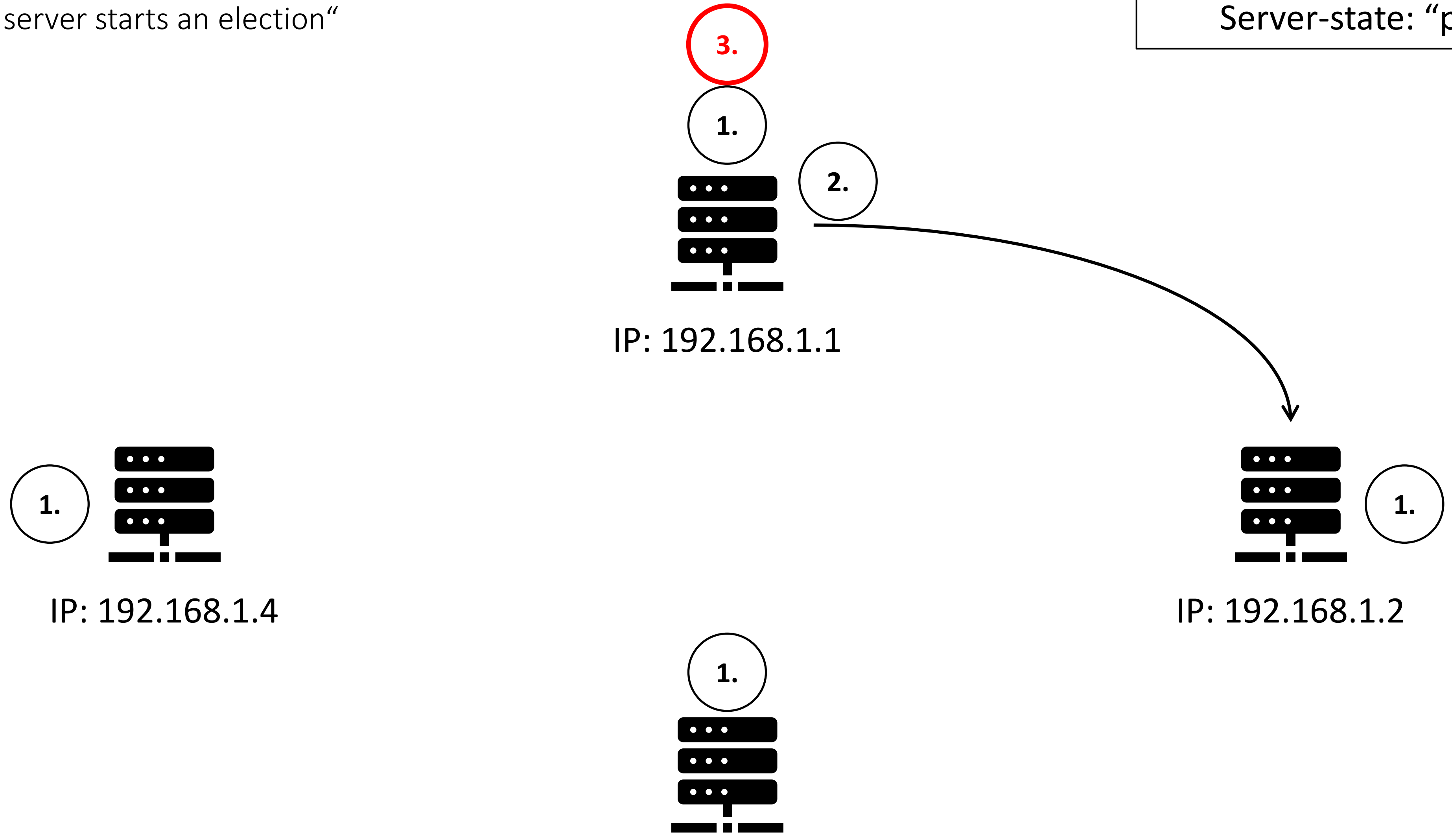
Creation of election message SE:
SE(pid, isLeader)
→ SE(192.168.1.1, False)
Send SE to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“

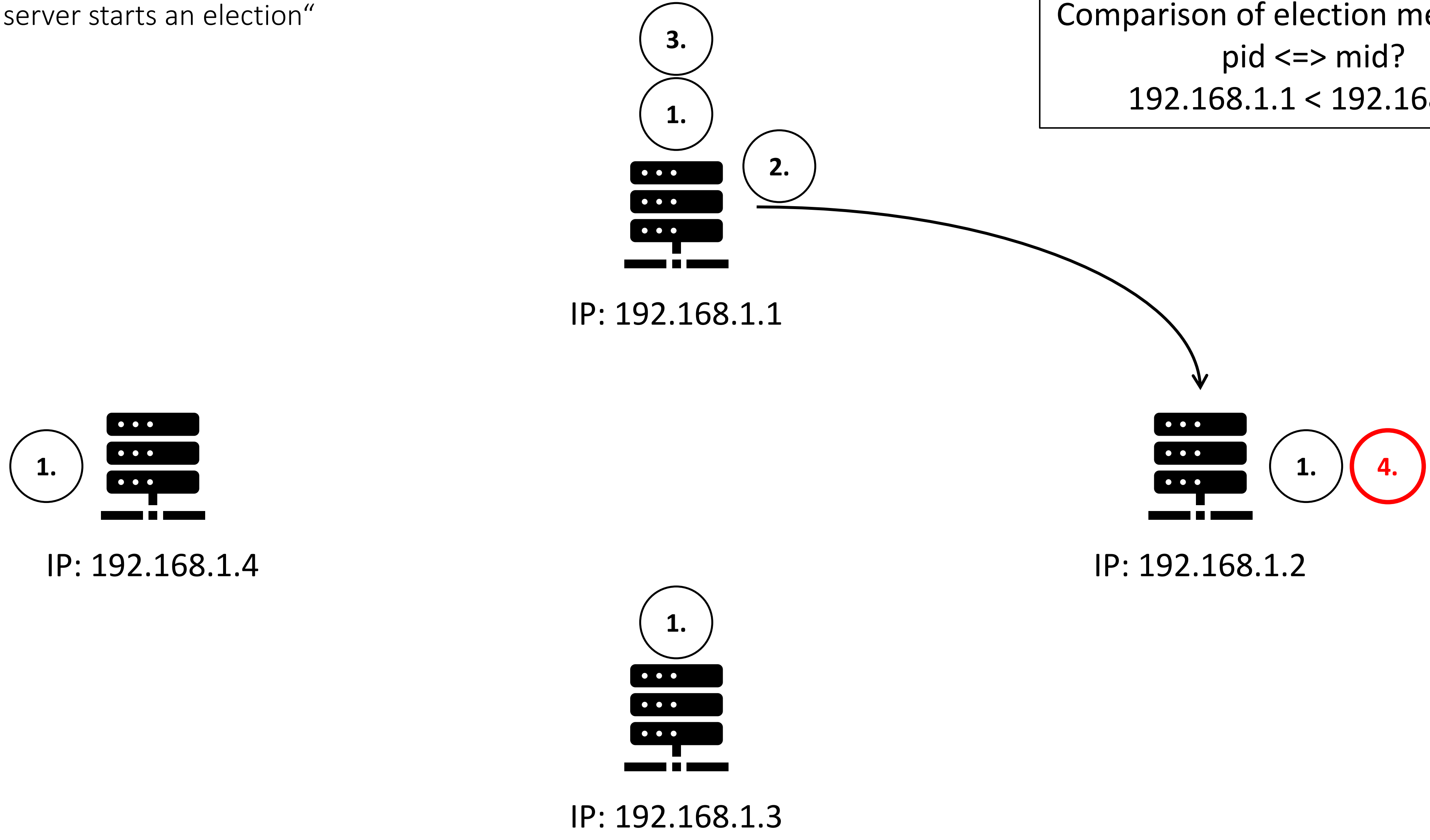
3\.

Server-state: “participant”



Chang and Roberts algorithm

„just one server starts an election“



4.

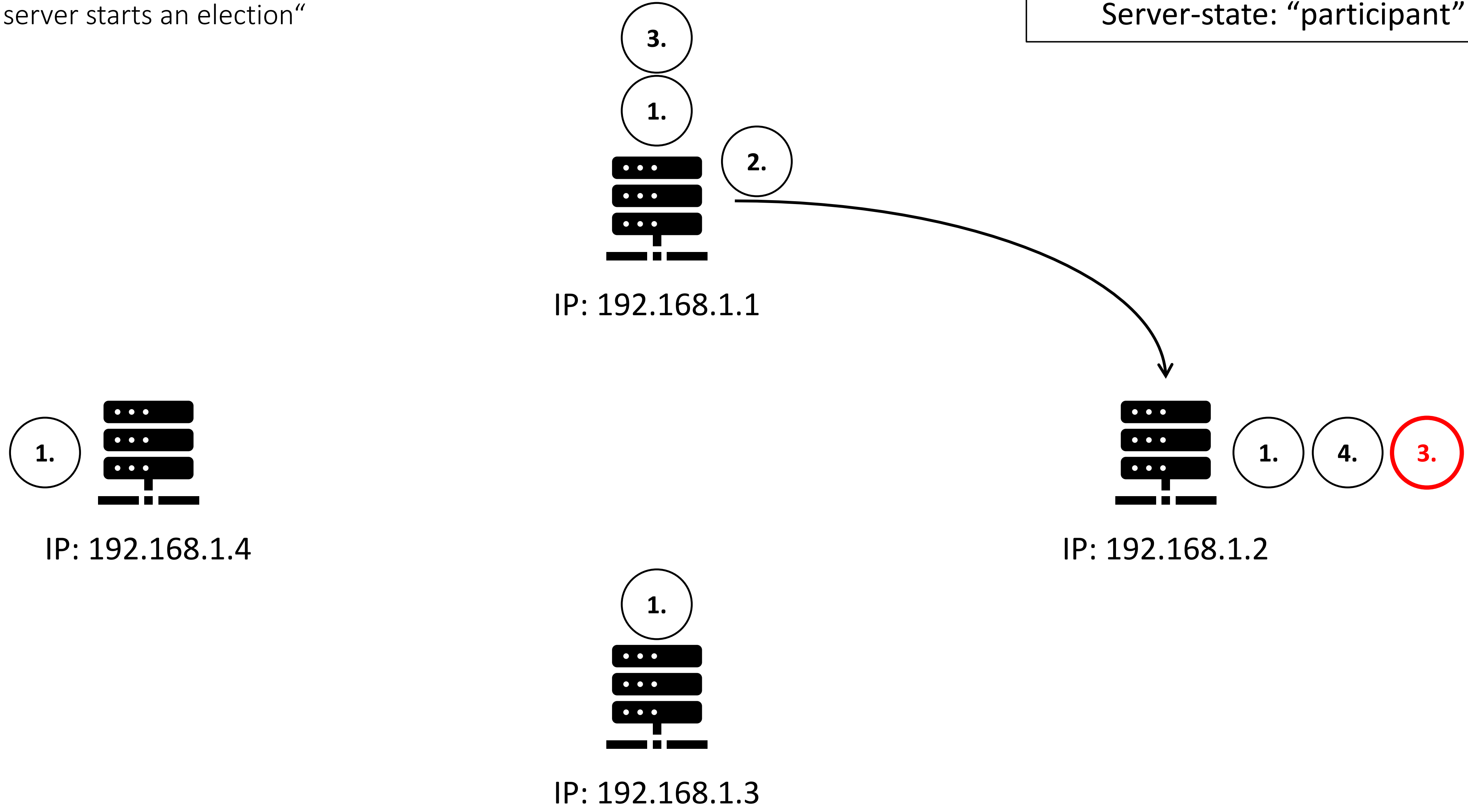
Comparison of election message SE:

pid <=> mid?

192.168.1.1 < 192.168.1.2

Chang and Roberts algorithm

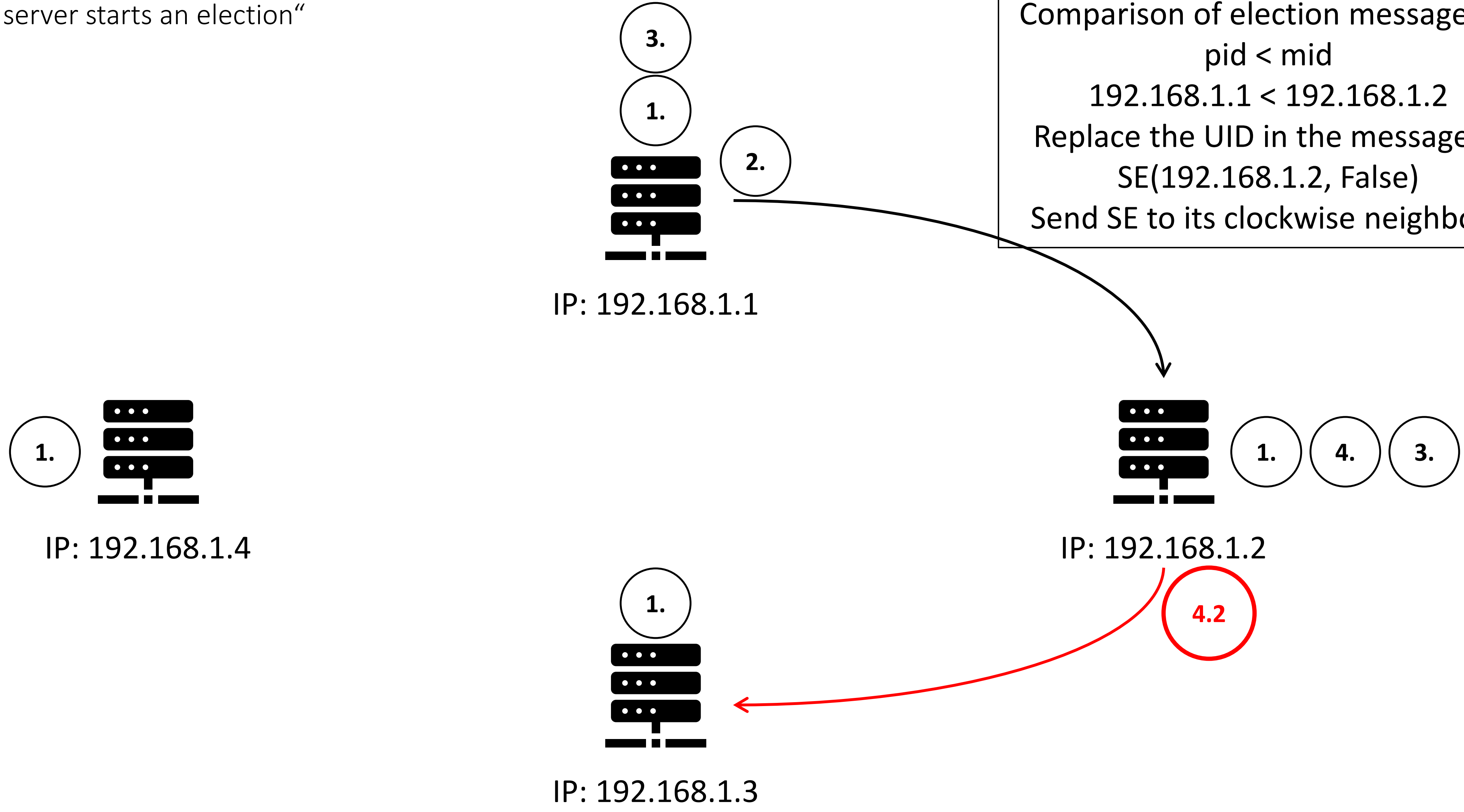
„just one server starts an election“



3.
Server-state: “participant”

Chang and Roberts algorithm

„just one server starts an election“



4.2

Comparison of election message SE:

pid < mid

192.168.1.1 < 192.168.1.2

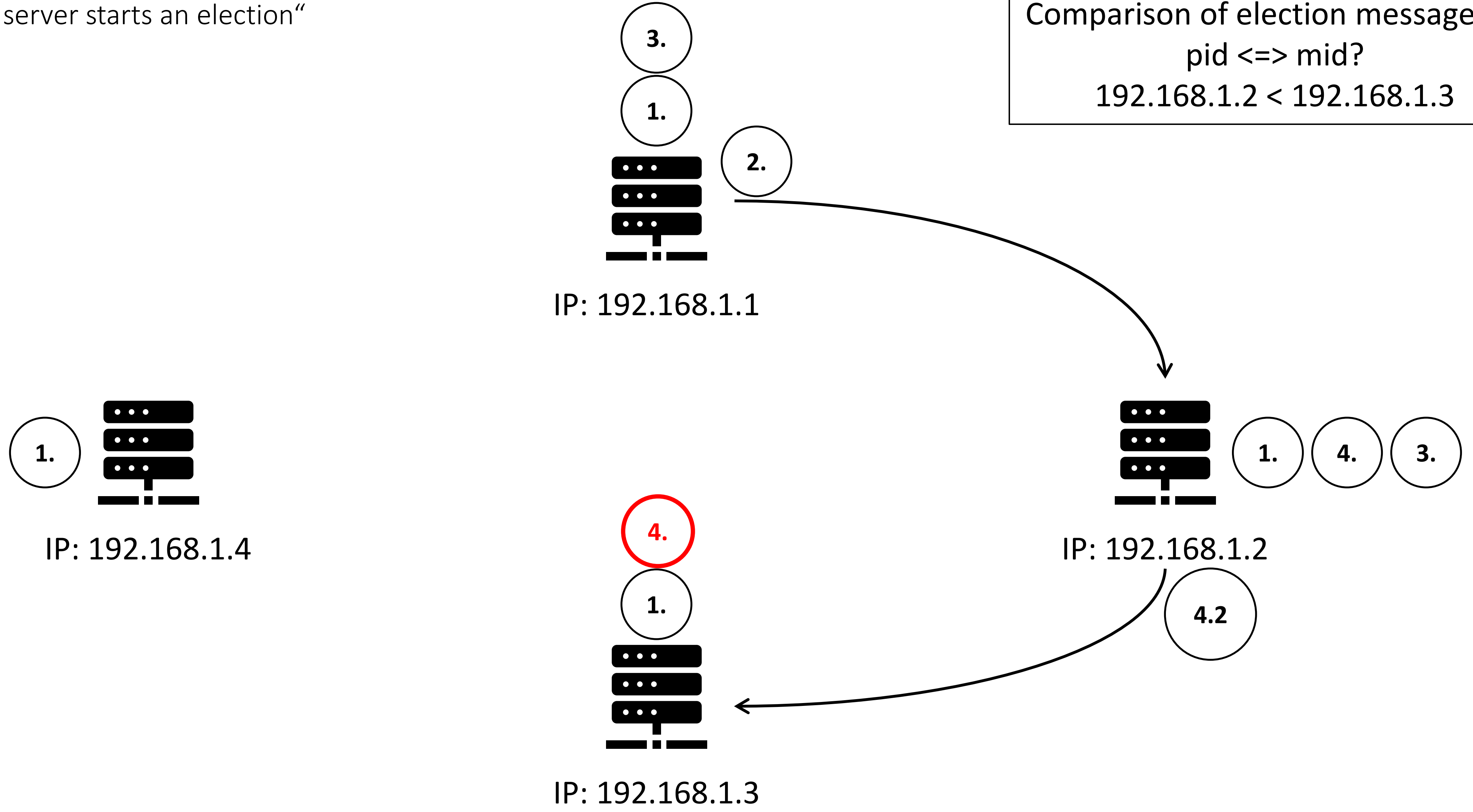
Replace the UID in the message E:

SE(192.168.1.2, False)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“



4.

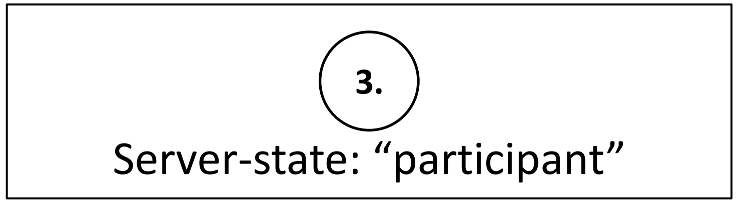
Comparison of election message SE:

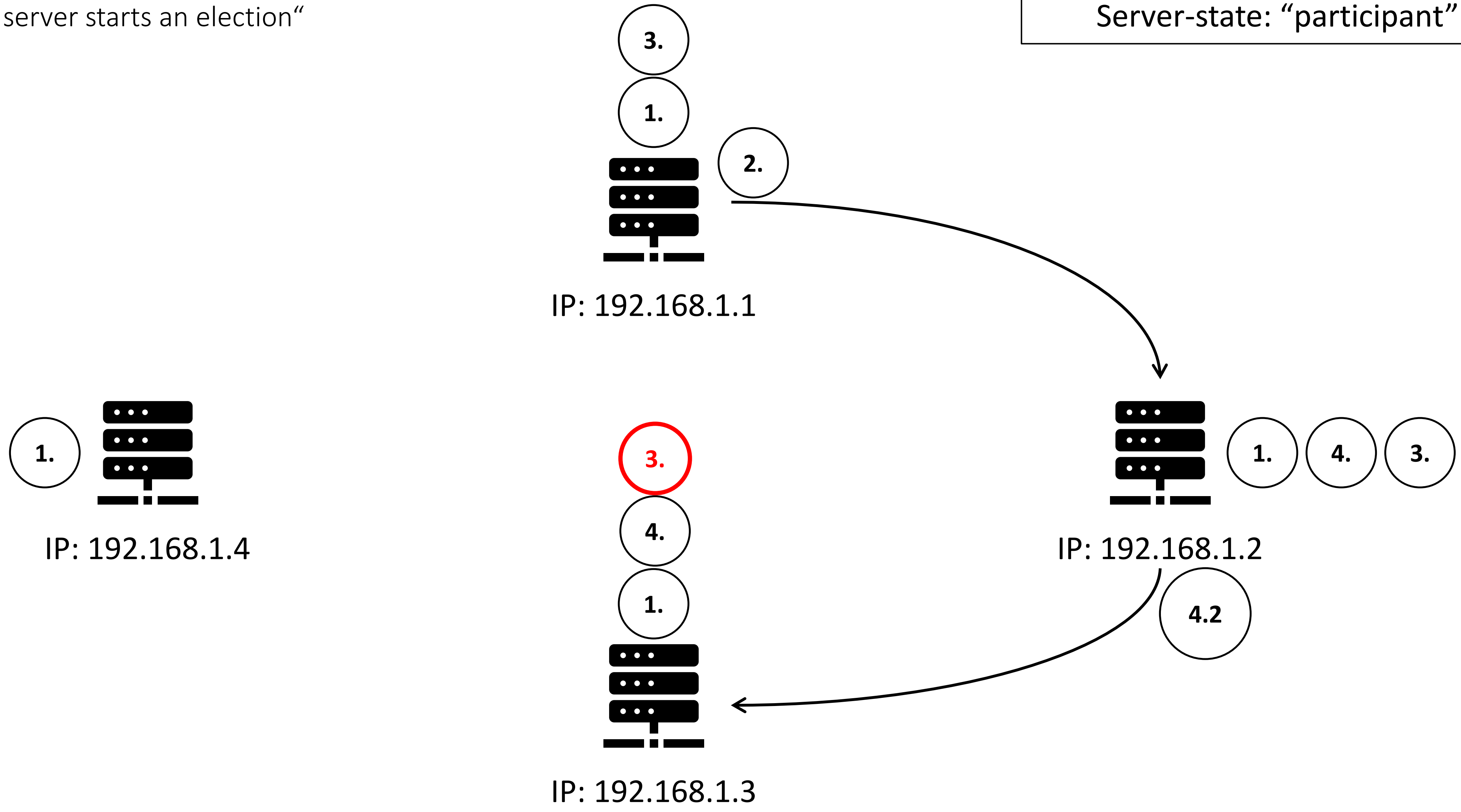
pid <=> mid?

192.168.1.2 < 192.168.1.3

Chang and Roberts algorithm

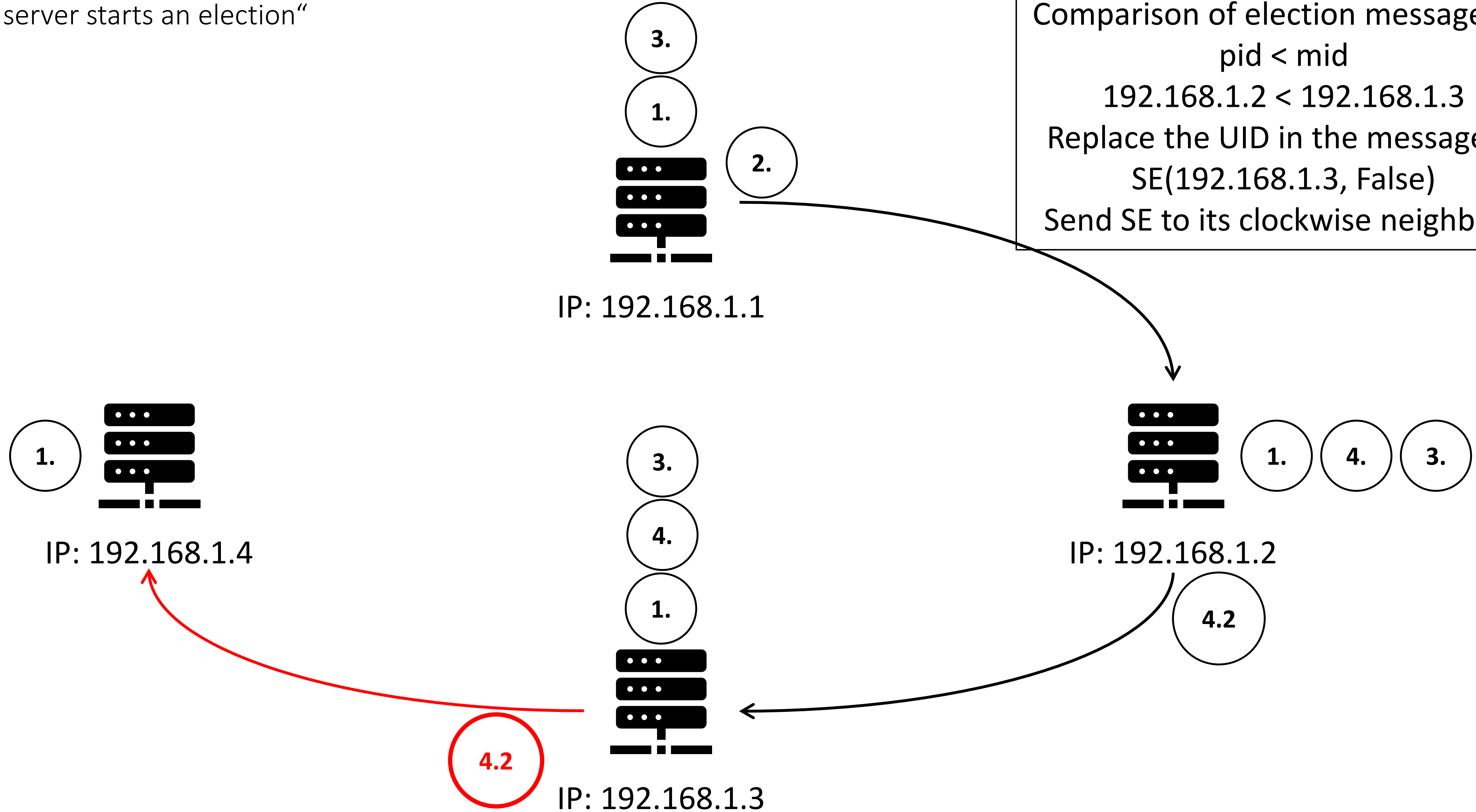
„just one server starts an election“





Chang and Roberts algorithm

„just one server starts an election“



4.2

Comparison of election message SE:

pid < mid

192.168.1.2 < 192.168.1.3

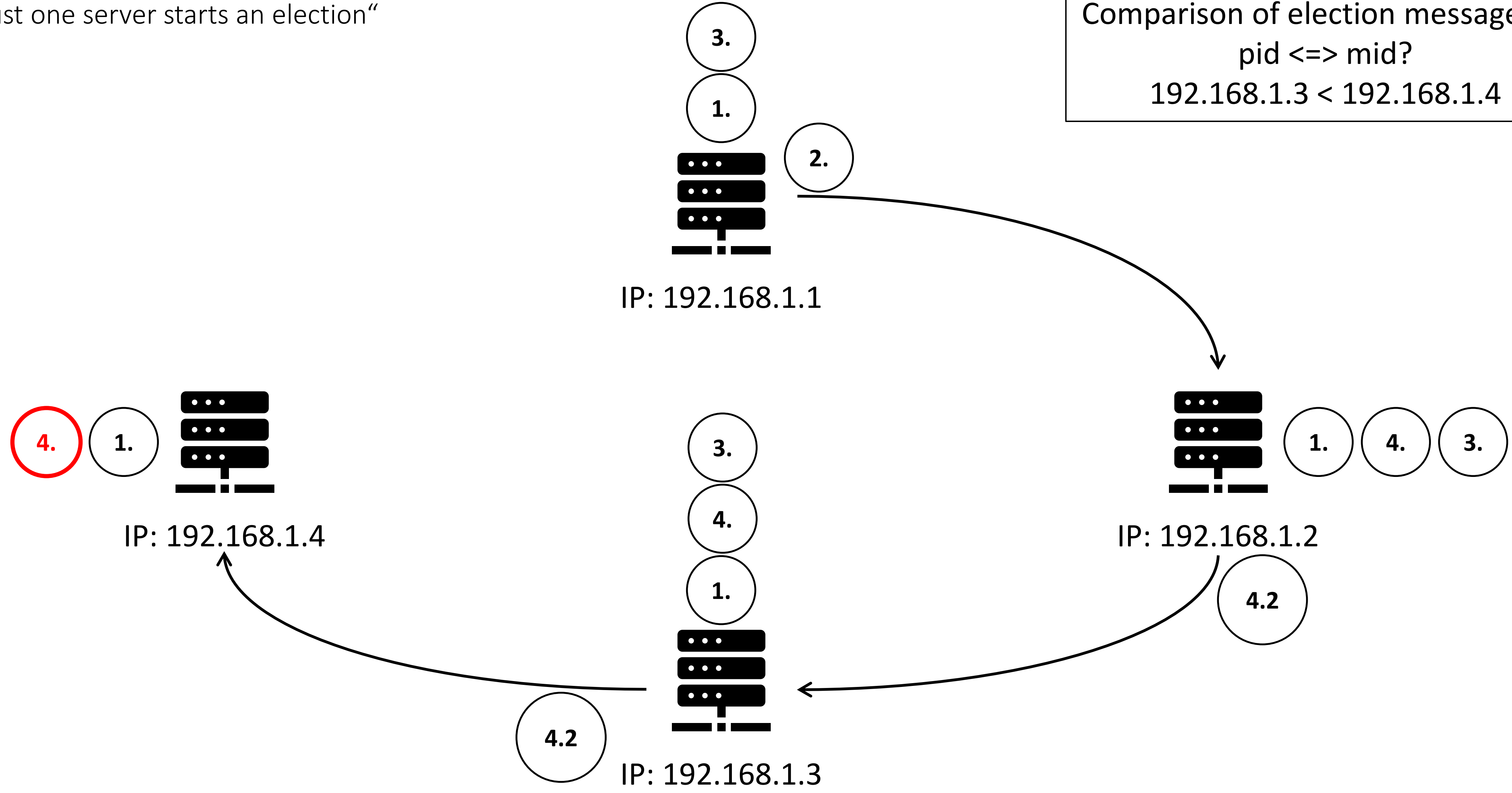
Replace the UID in the message E:

SE(192.168.1.3, False)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“



4.

Comparison of election message SE:

pid <=> mid?

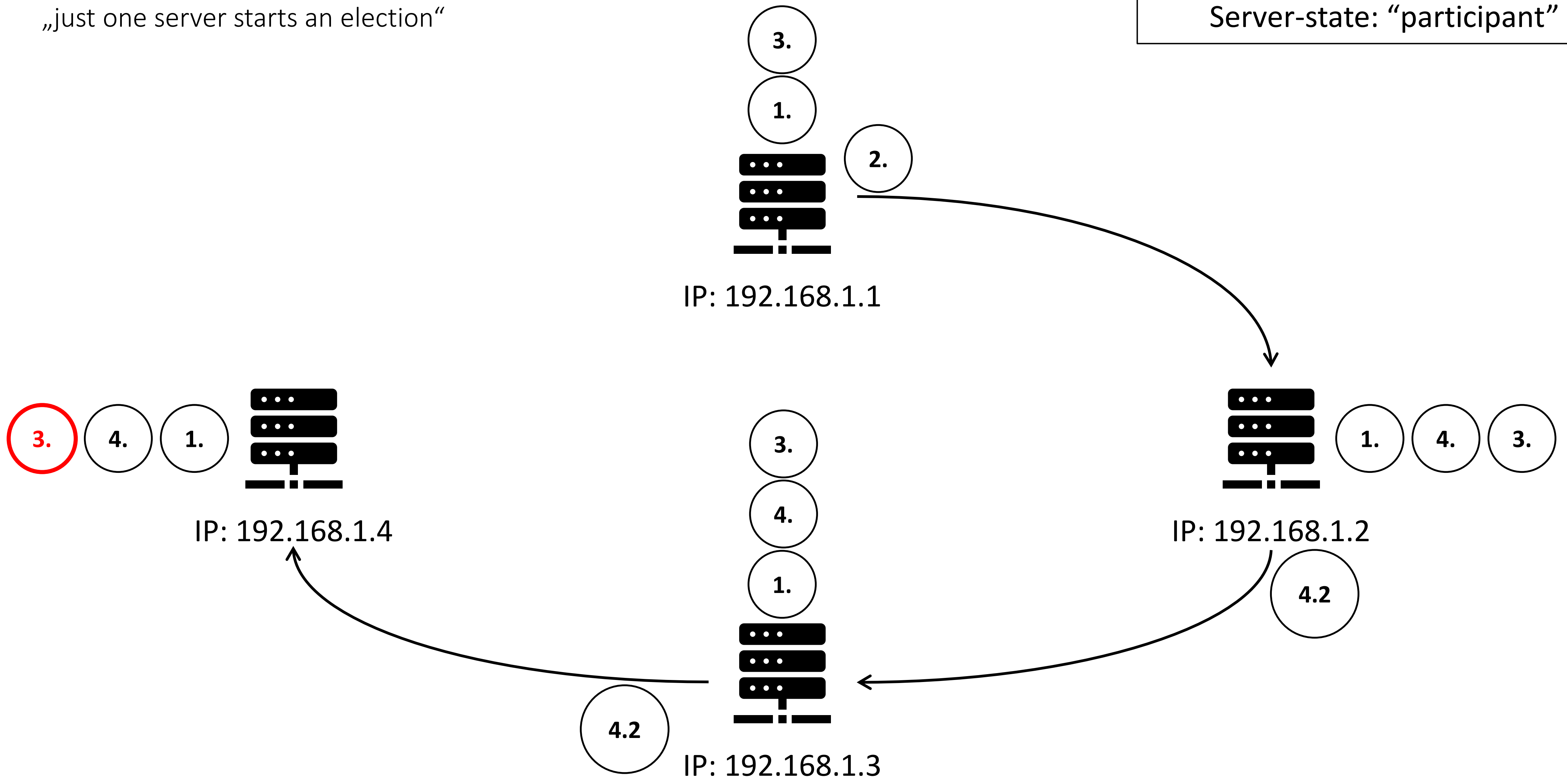
192.168.1.3 < 192.168.1.4

Chang and Roberts algorithm

„just one server starts an election“

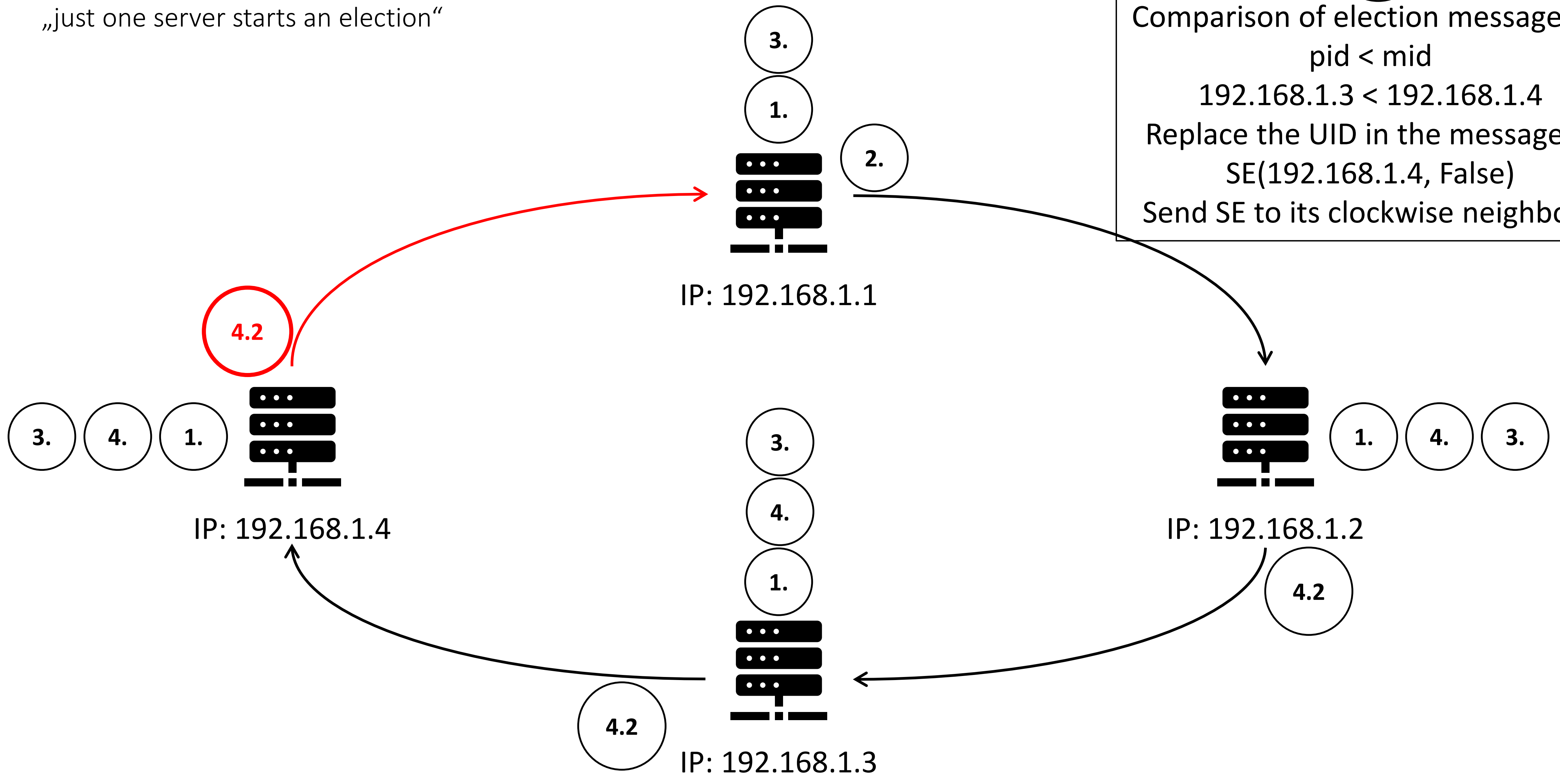
3.

Server-state: “participant”



Chang and Roberts algorithm

„just one server starts an election“



4.2

Comparison of election message SE:

pid < mid

192.168.1.3 < 192.168.1.4

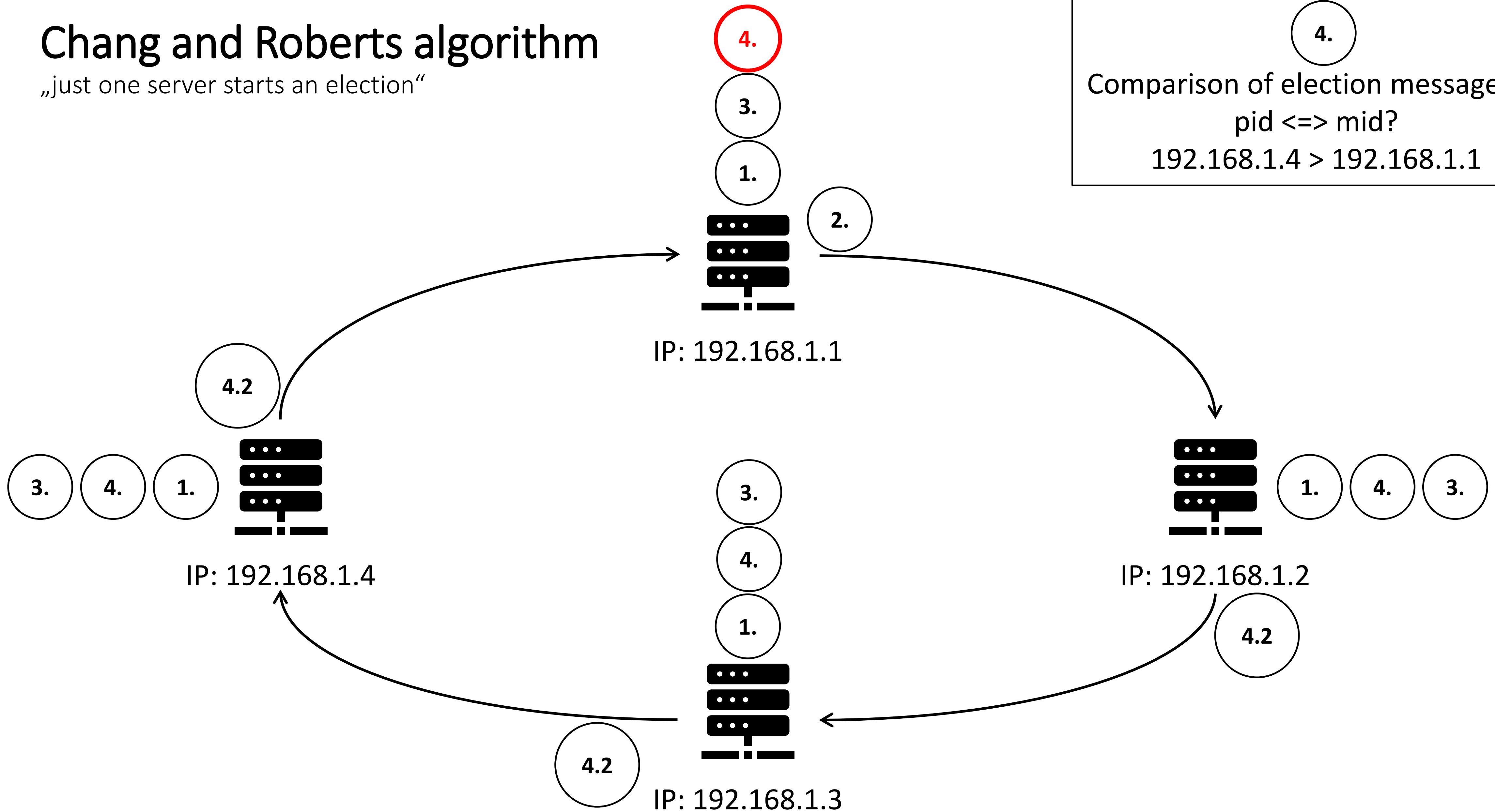
Replace the UID in the message E:

SE(192.168.1.4, False)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“



4.

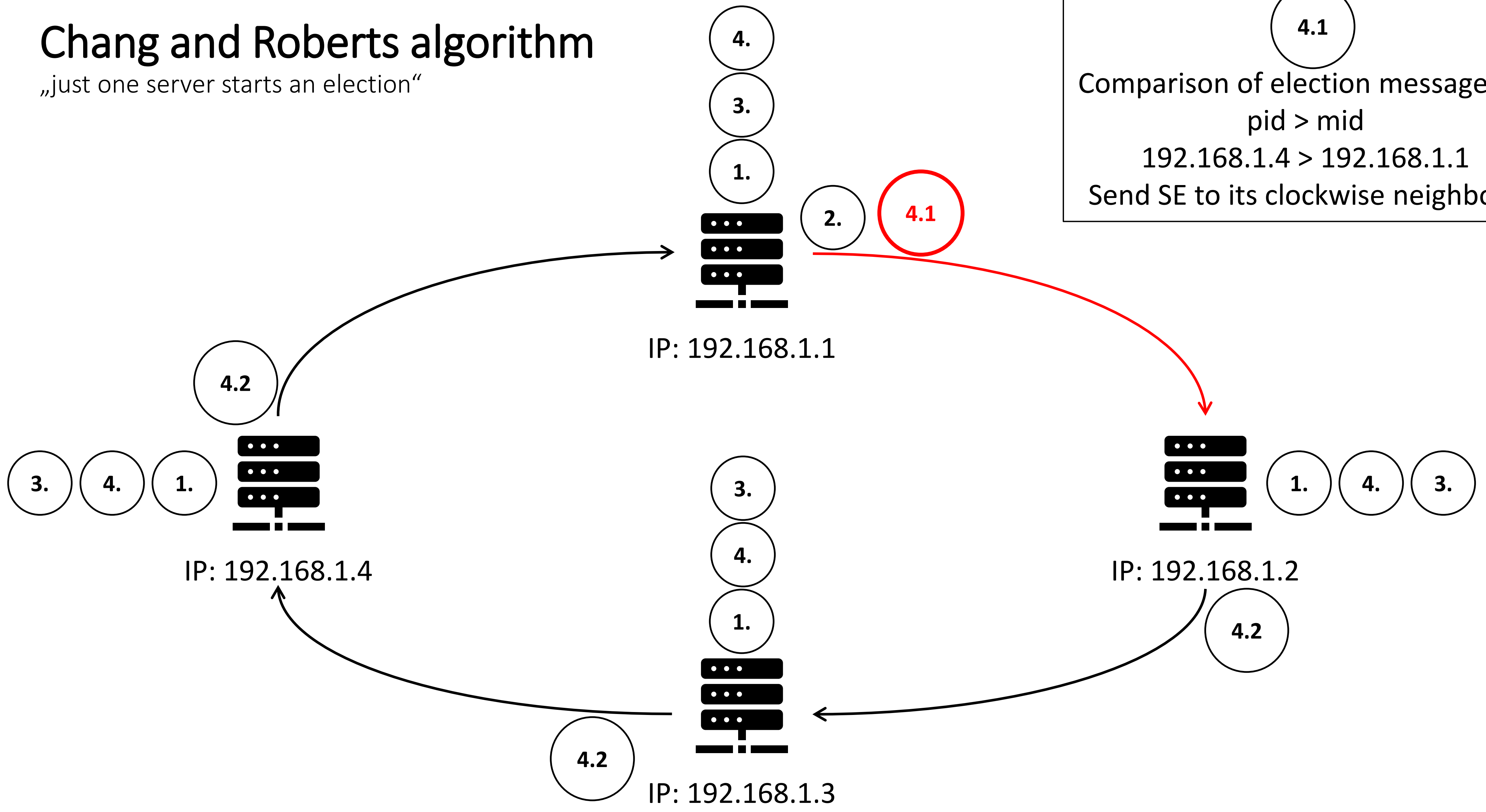
Comparison of election message SE:

pid <=> mid?

192.168.1.4 > 192.168.1.1

Chang and Roberts algorithm

„just one server starts an election“



4.1

Comparison of election message SE:

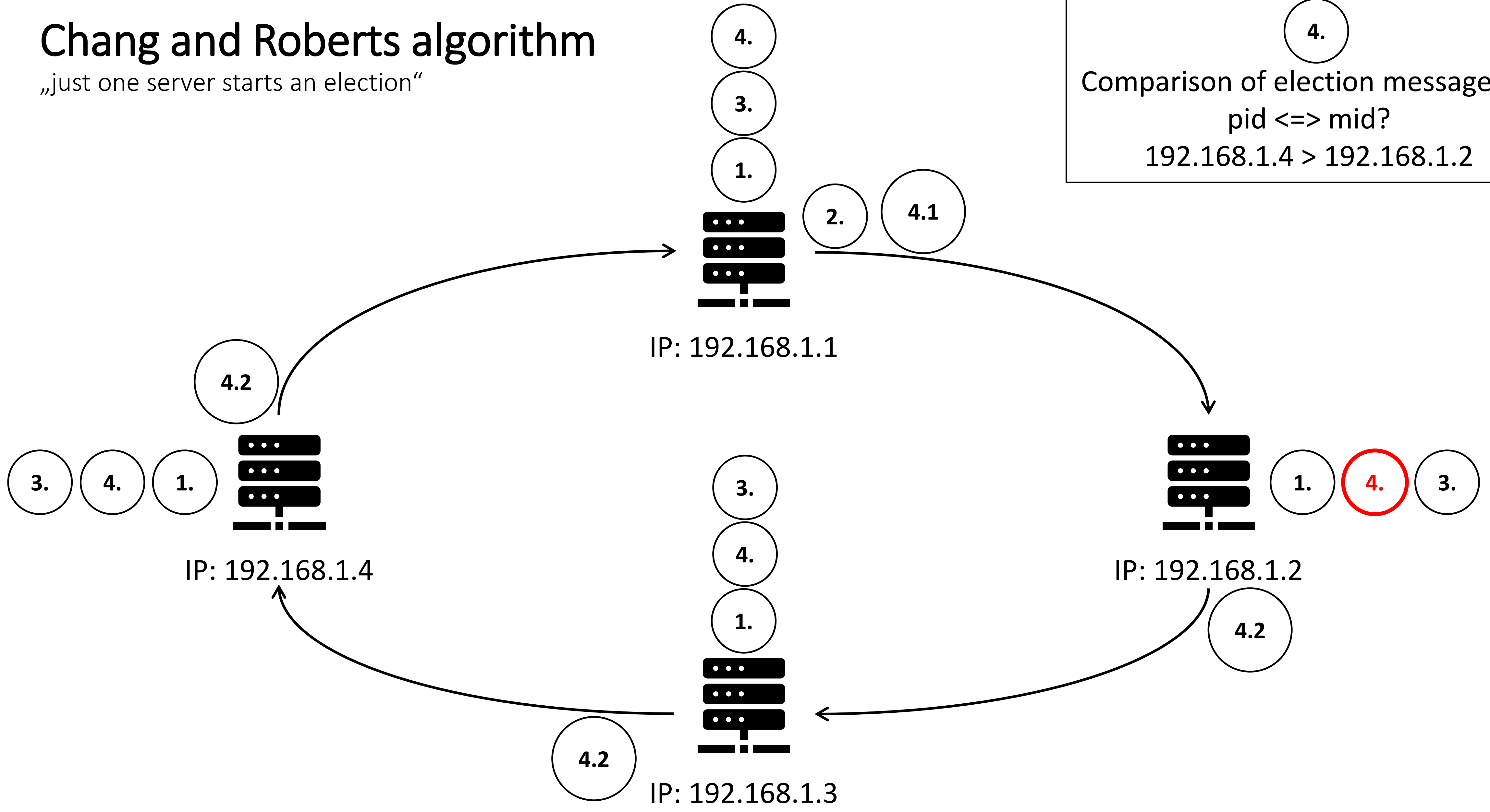
pid > mid

192.168.1.4 > 192.168.1.1

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“



4.

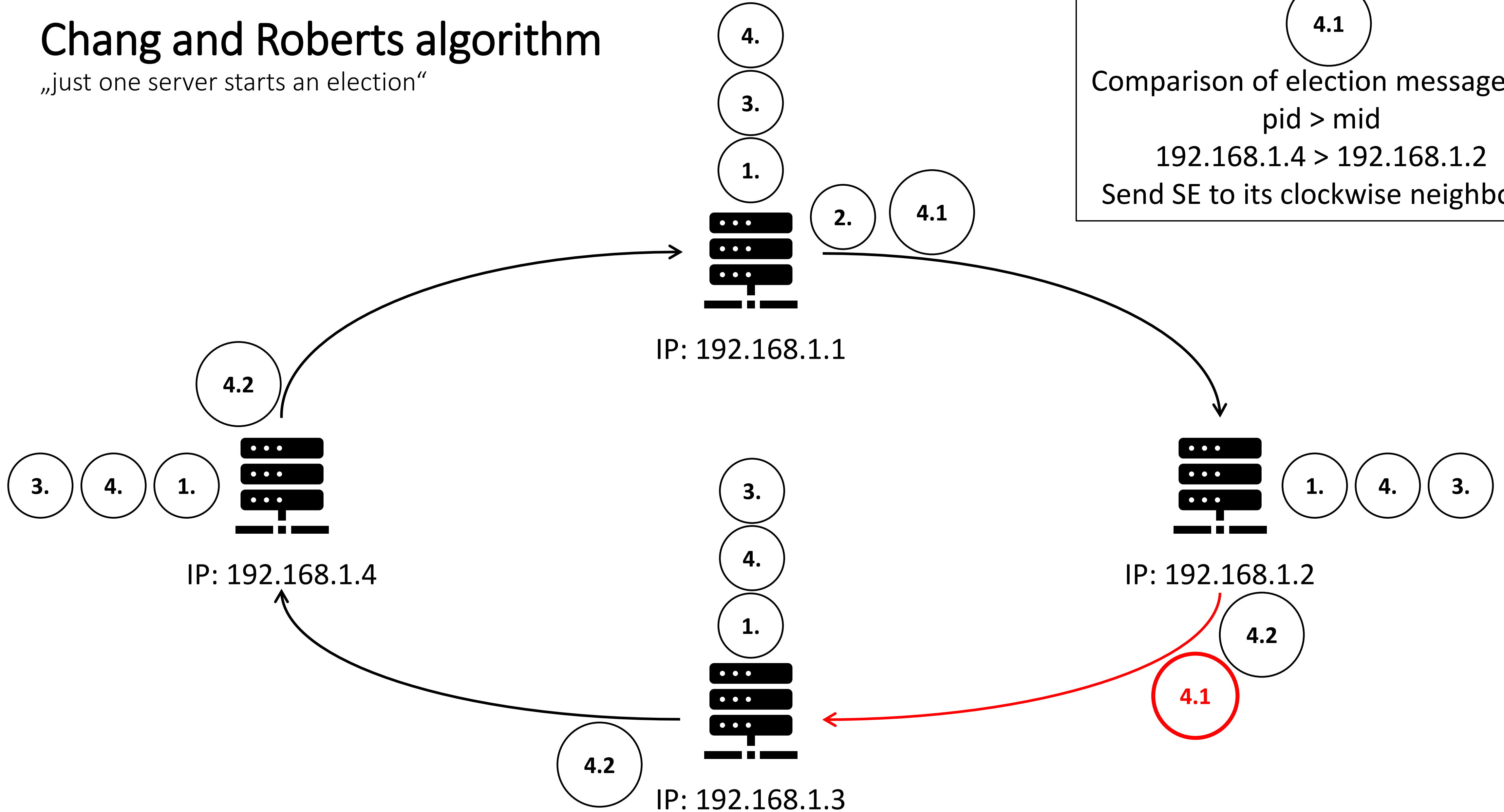
Comparison of election message SE:

pid <=> mid?

192.168.1.4 > 192.168.1.2

Chang and Roberts algorithm

„just one server starts an election“



4.1

Comparison of election message SE:

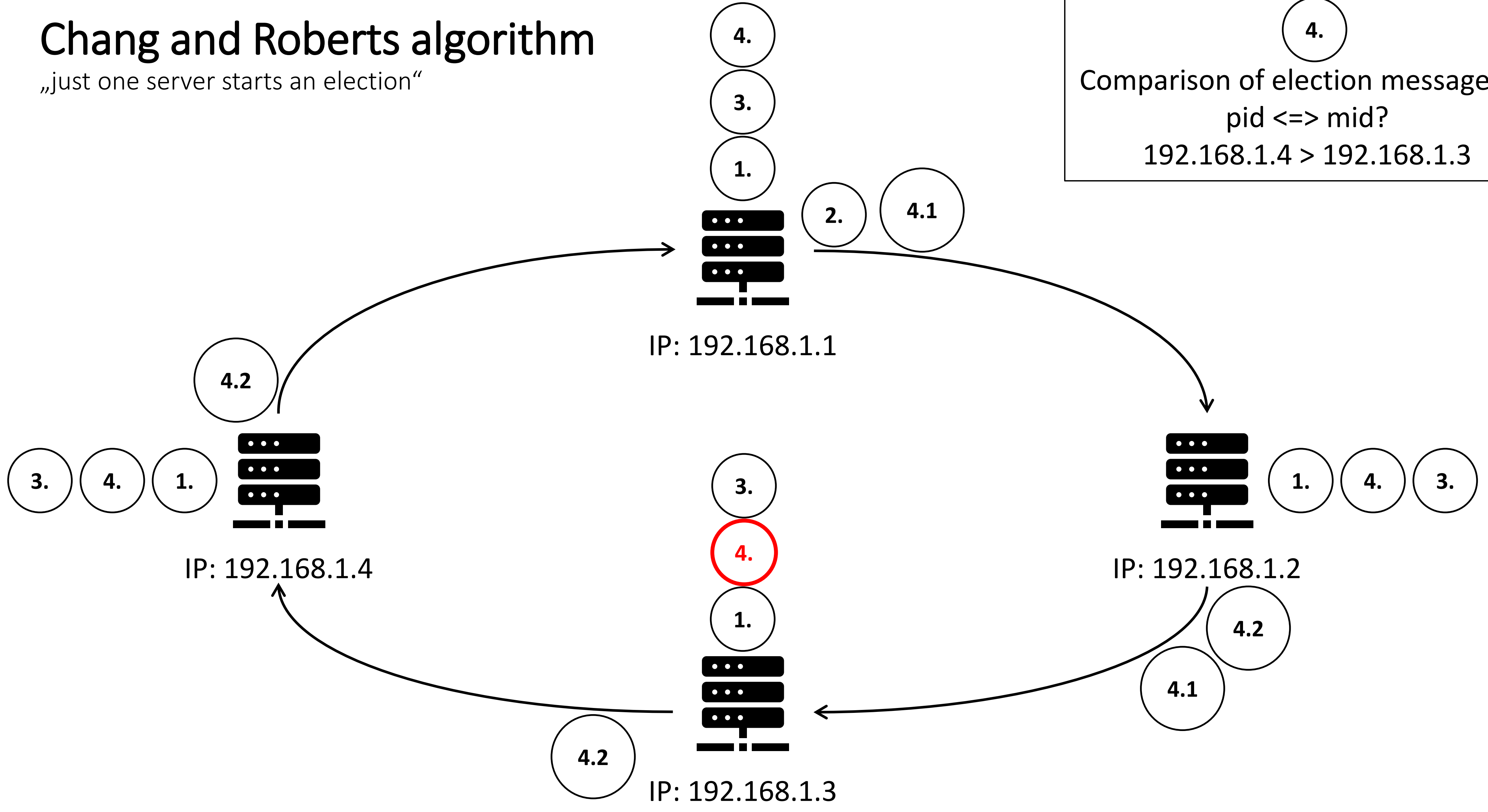
pid > mid

192.168.1.4 > 192.168.1.2

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“



4.

Comparison of election message SE:

pid <=> mid?

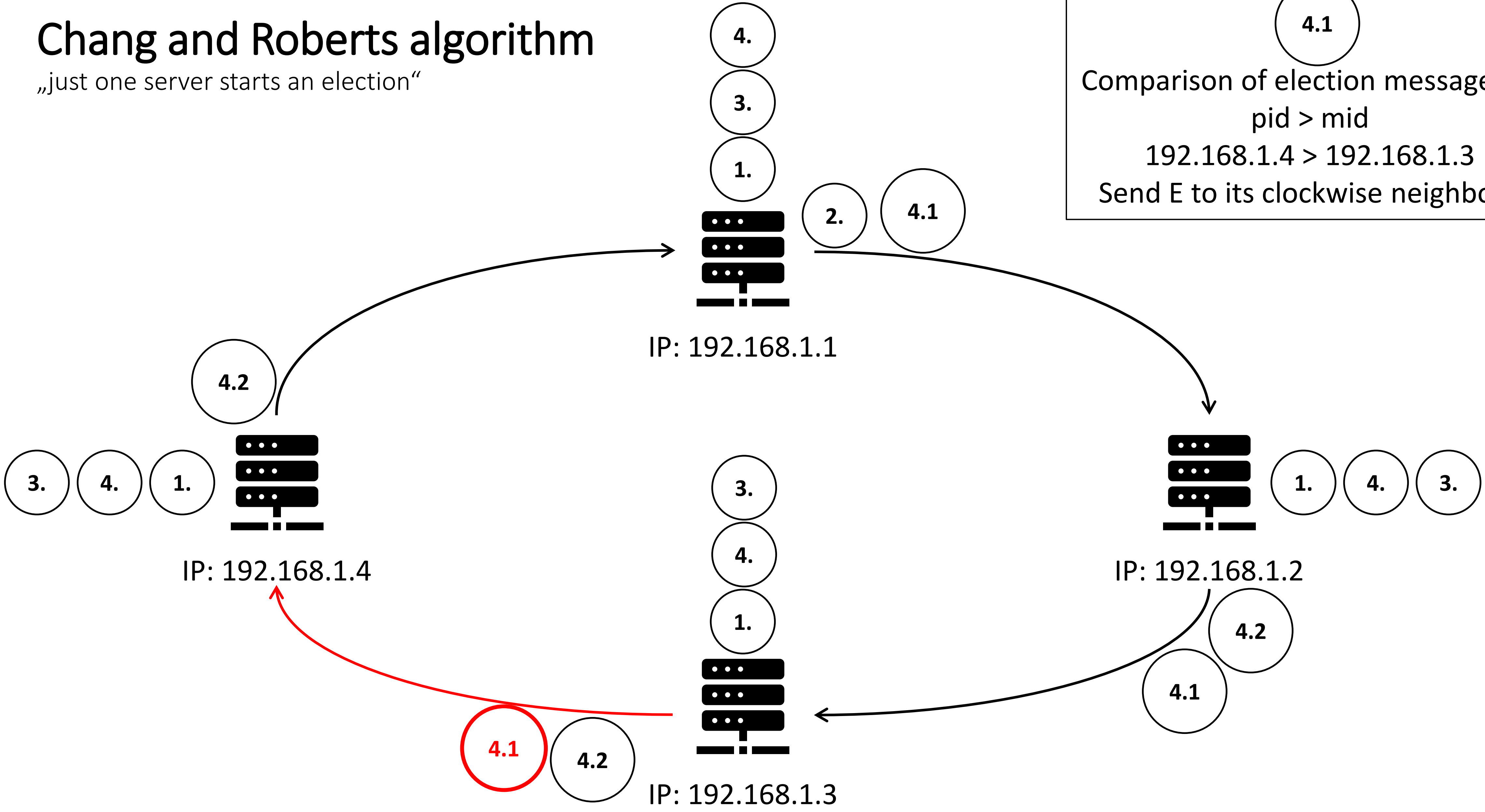
192.168.1.4 > 192.168.1.3

Chang and Roberts algorithm

„just one server starts an election“

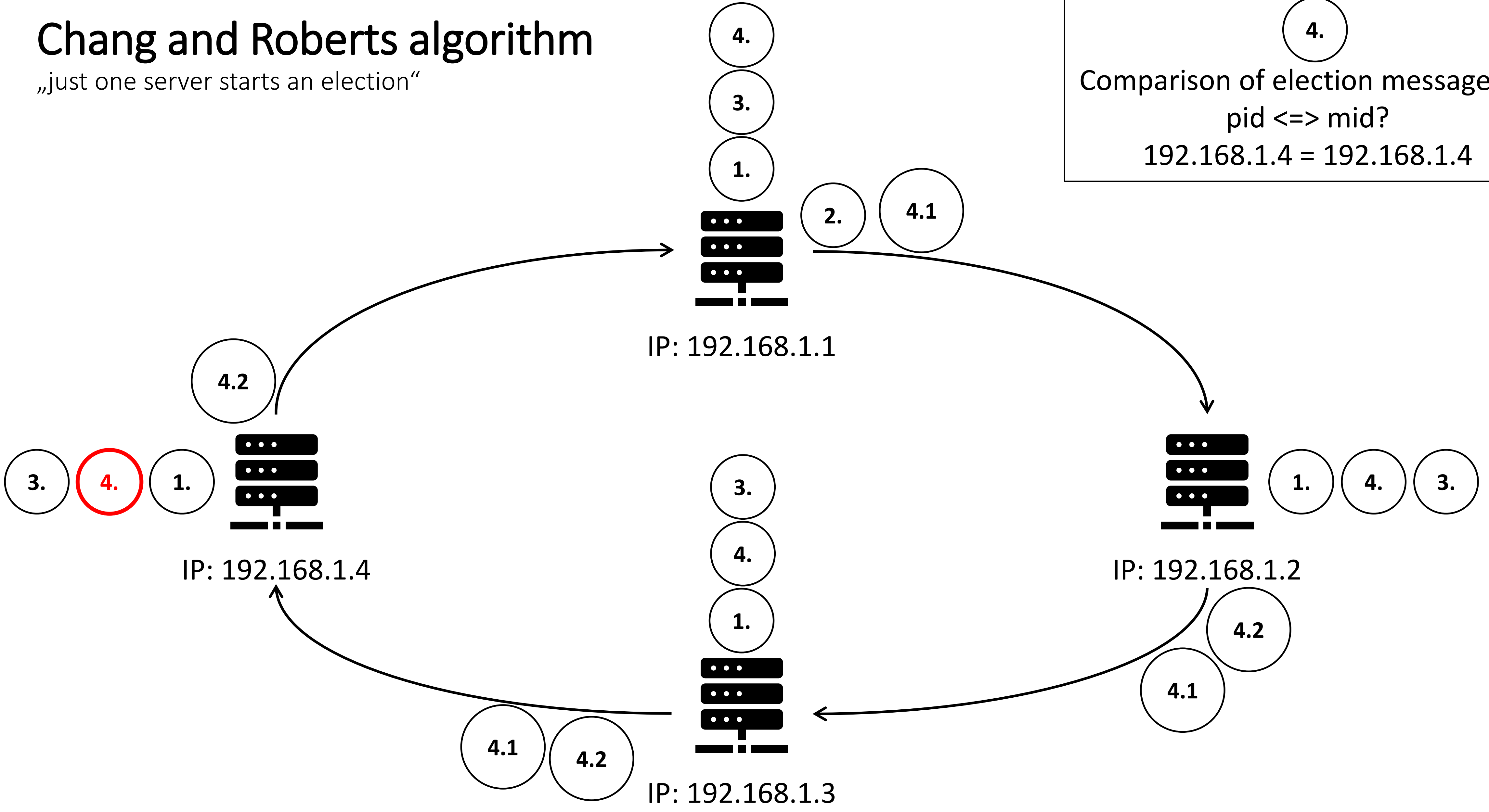
4.1

Comparison of election message SE:
pid > mid
192.168.1.4 > 192.168.1.3
Send E to its clockwise neighbour



Chang and Roberts algorithm

„just one server starts an election“



4.

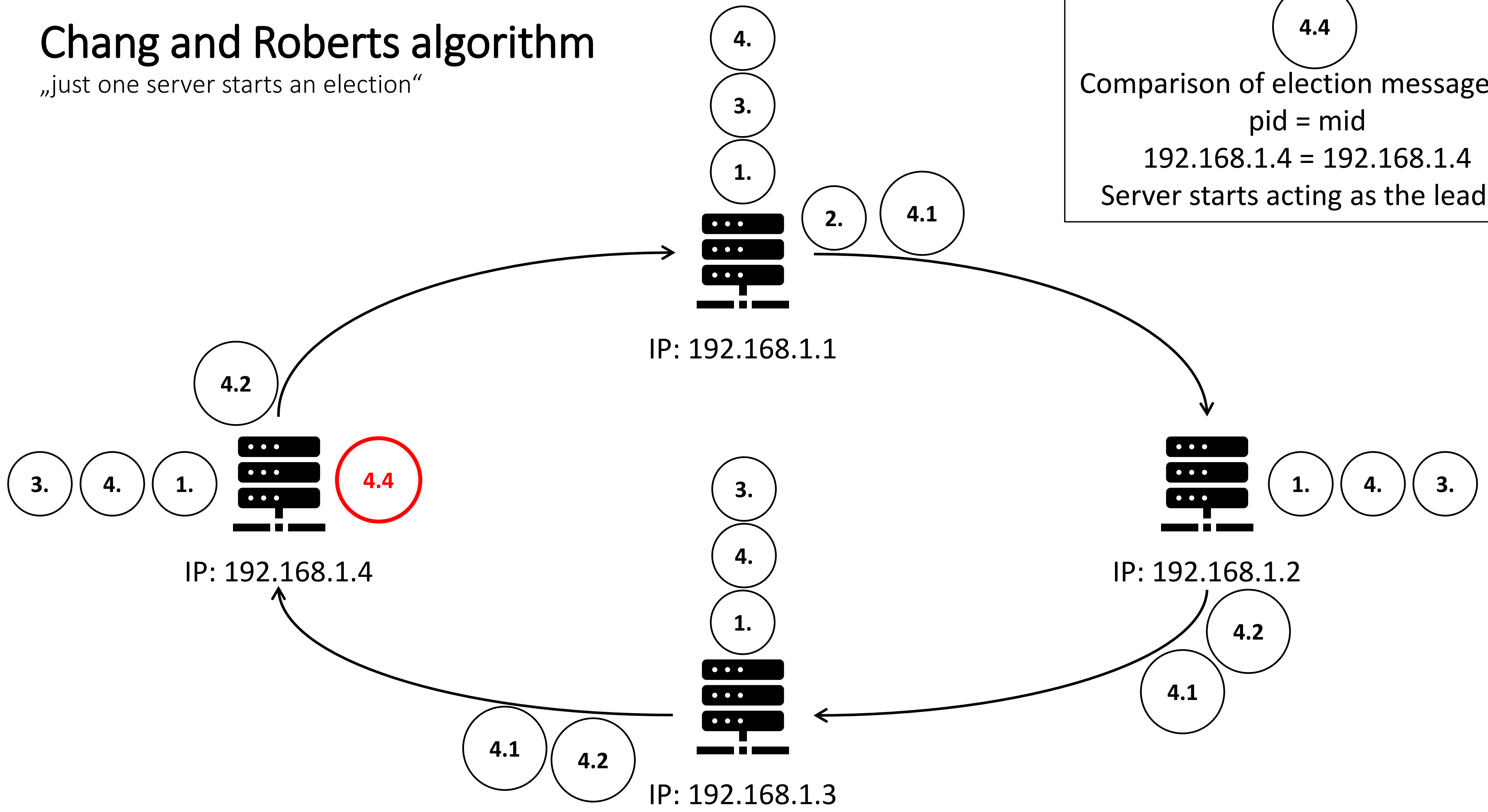
Comparison of election message SE:

pid <=> mid?

192.168.1.4 = 192.168.1.4

Chang and Roberts algorithm

„just one server starts an election“



4.4

Comparison of election message SE:

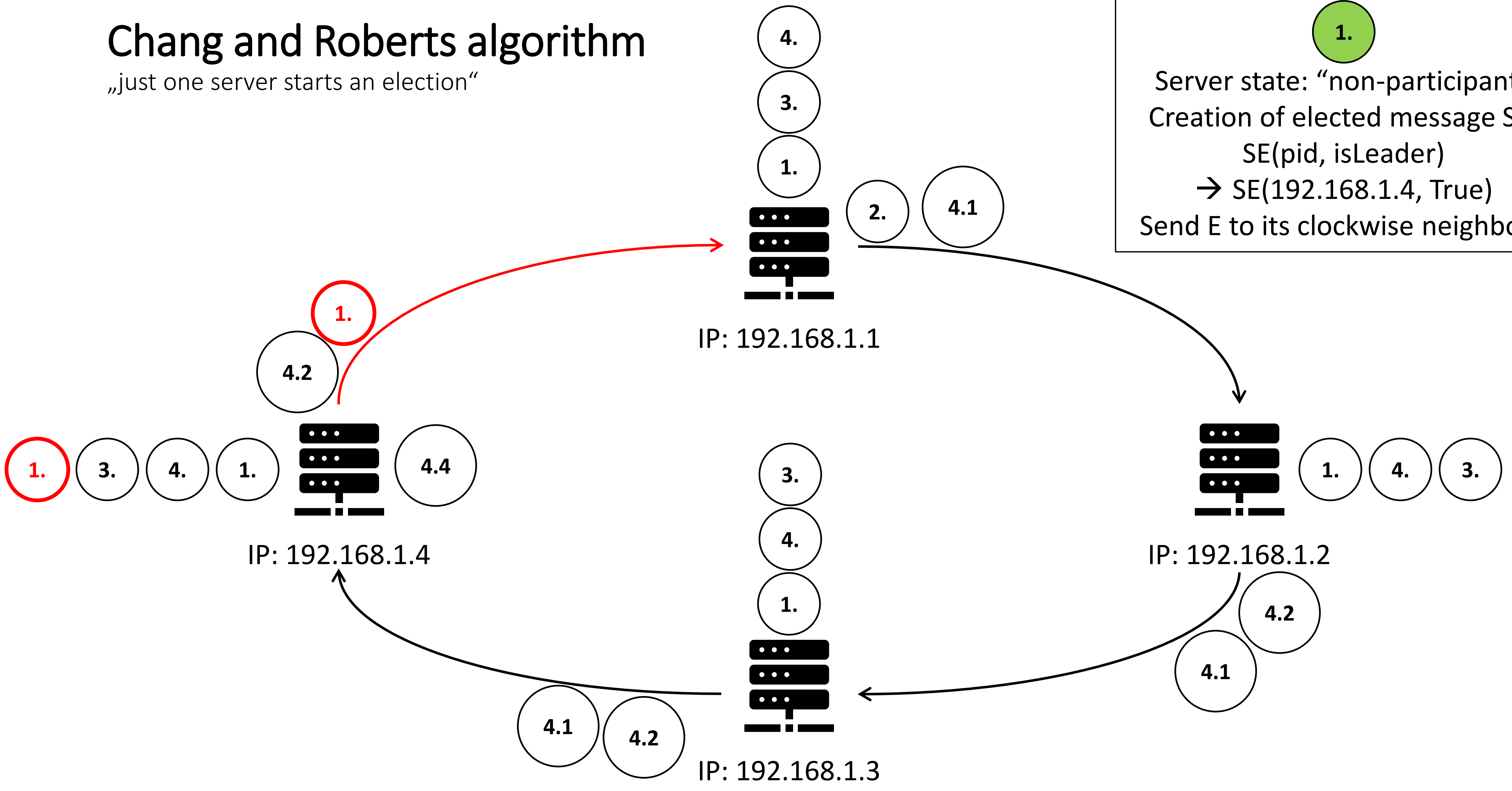
pid = mid

192.168.1.4 = 192.168.1.4

Server starts acting as the leader

Chang and Roberts algorithm

„just one server starts an election“

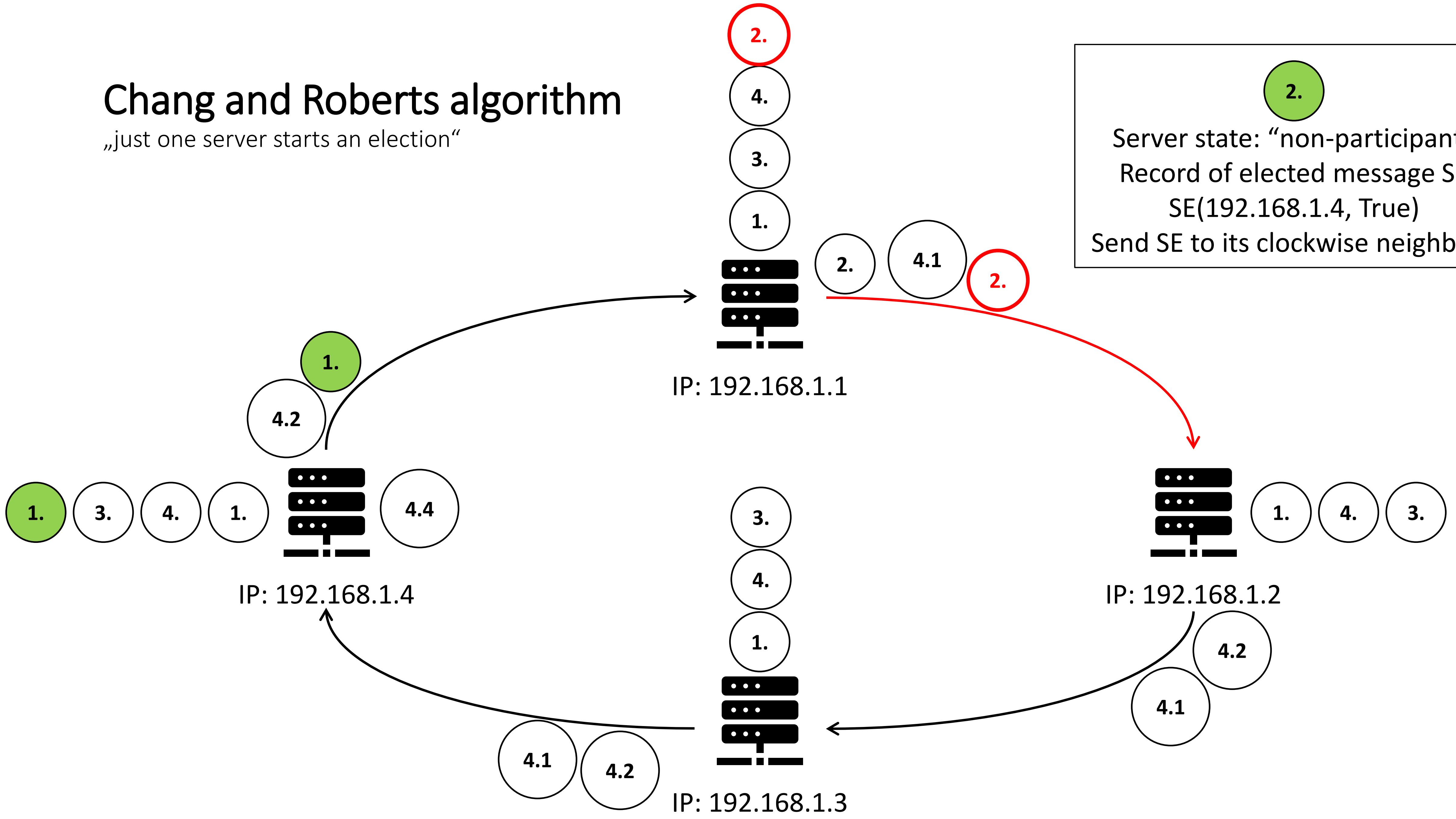


1.

Server state: “non-participant”
Creation of elected message SE:
SE(pid, isLeader)
→ SE(192.168.1.4, True)
Send E to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“



2.

Server state: “non-participant”

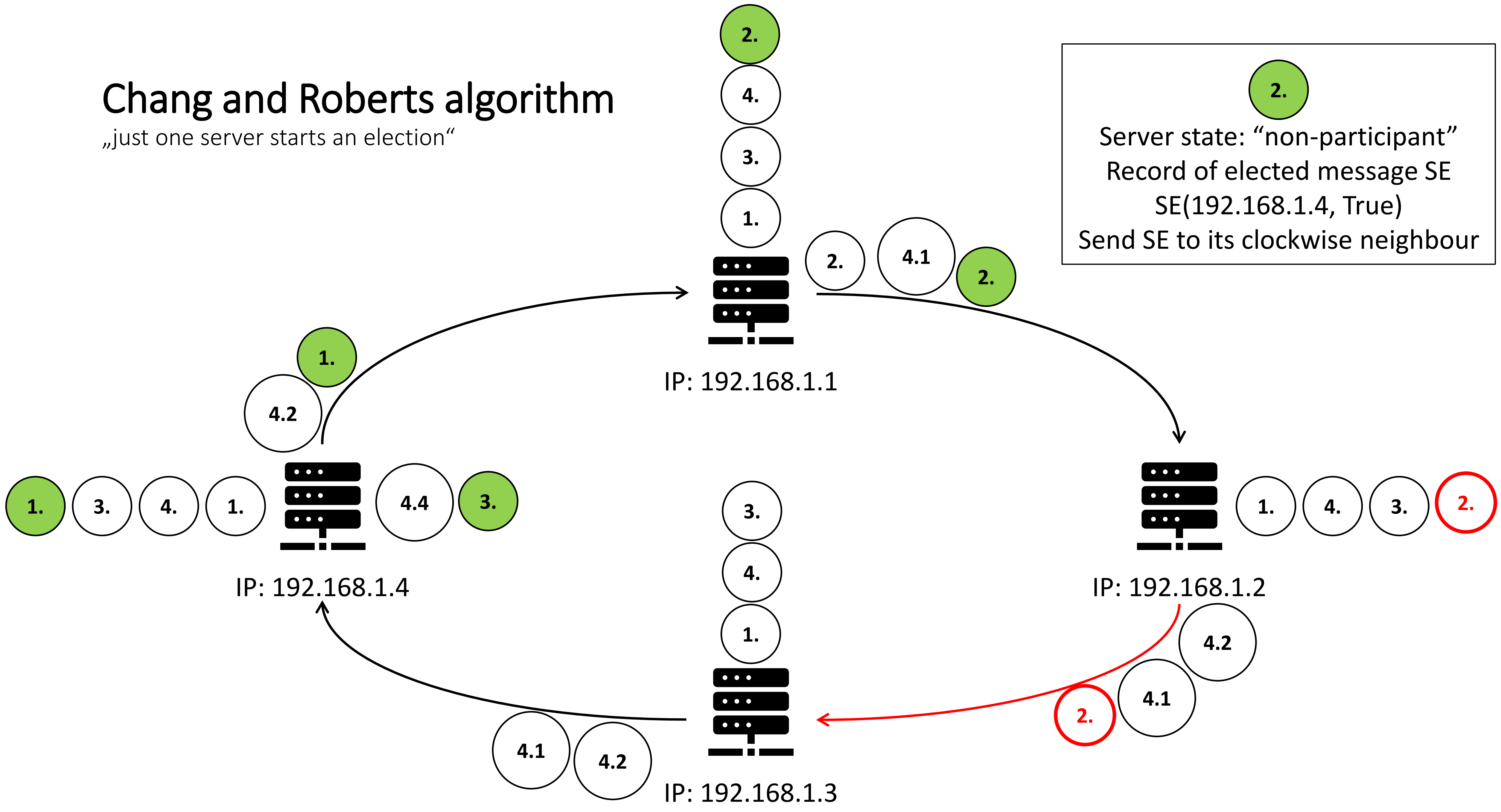
Record of elected message SE

SE(192.168.1.4, True)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“



2.

Server state: “non-participant”

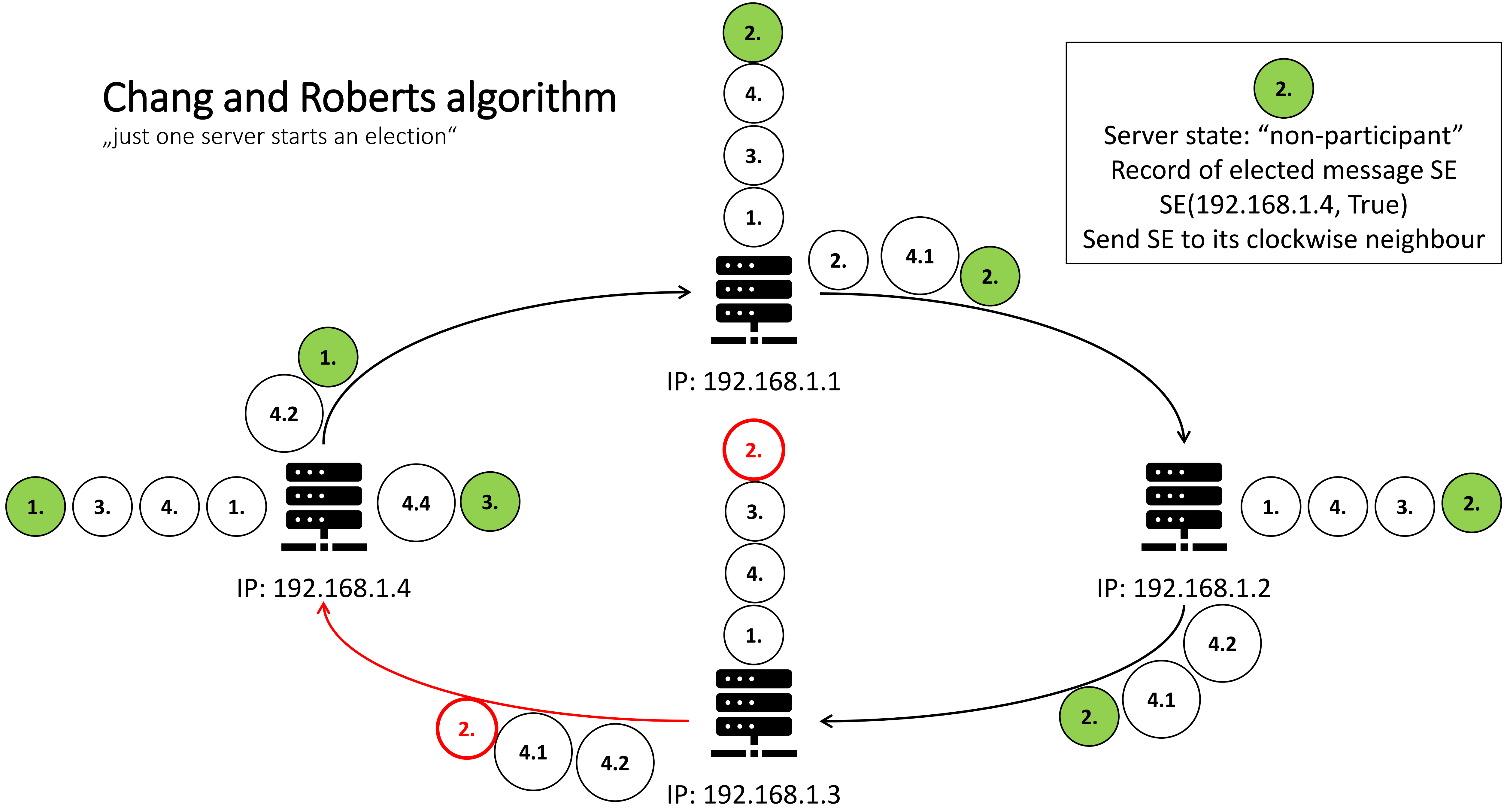
Record of elected message SE

SE(192.168.1.4, True)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“



2.

Server state: “non-participant”

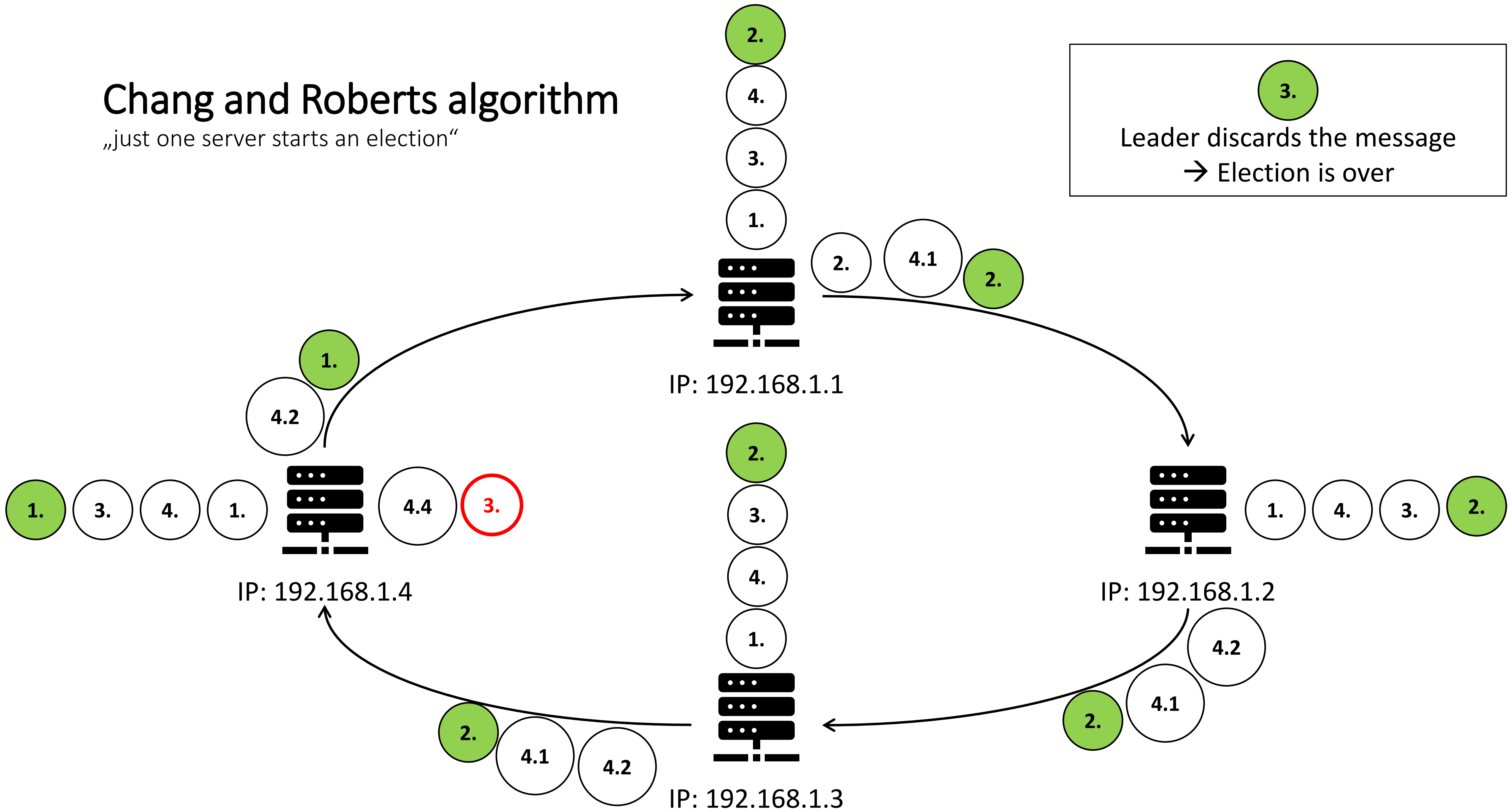
Record of elected message SE

SE(192.168.1.4, True)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„just one server starts an election“

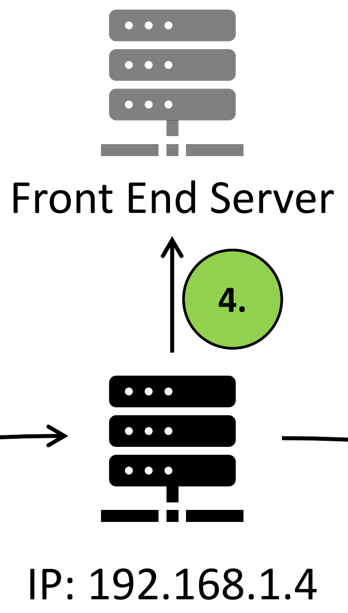


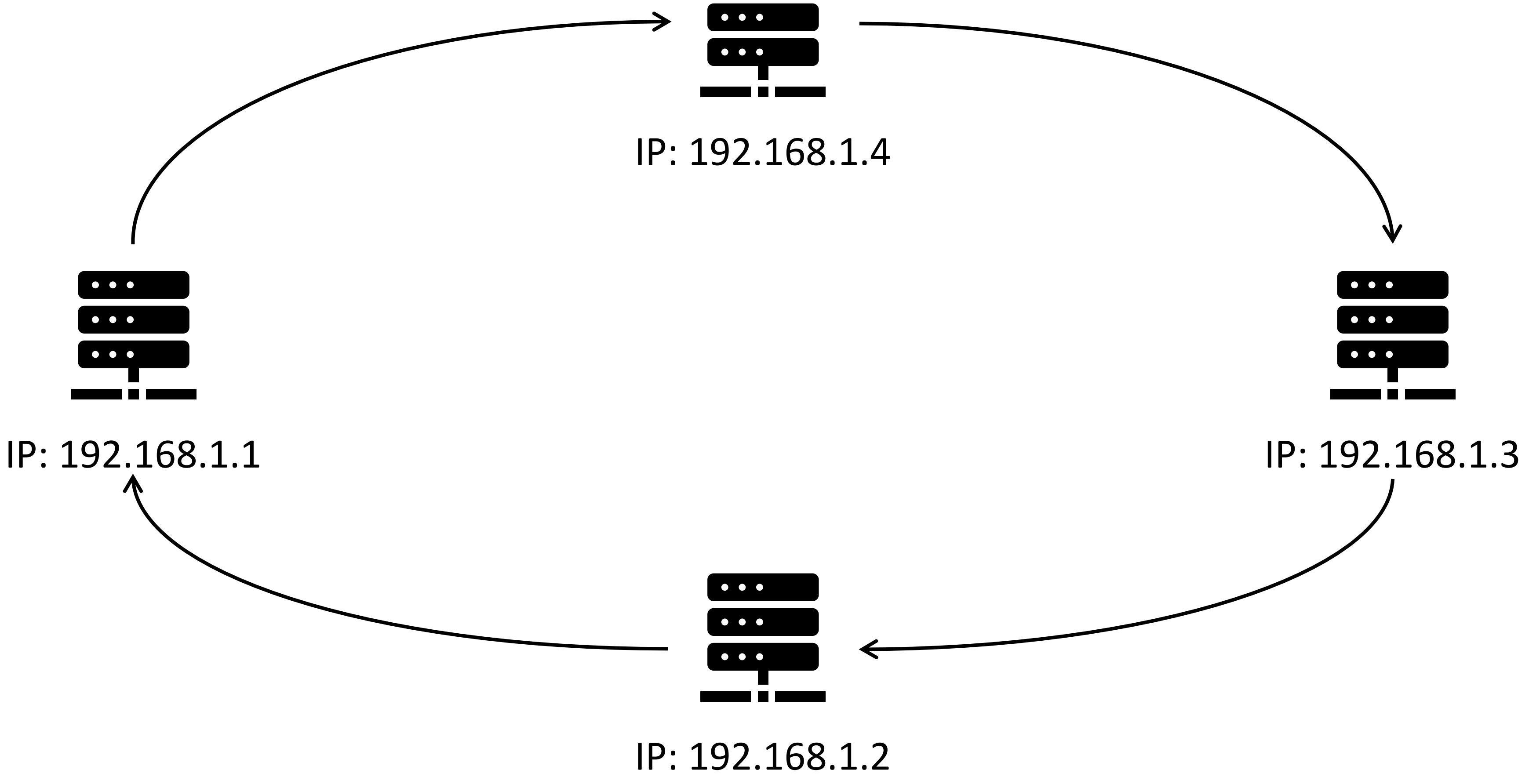
Leader discards the message

→ Election is over

Chang and Roberts algorithm

„just one server starts an election“





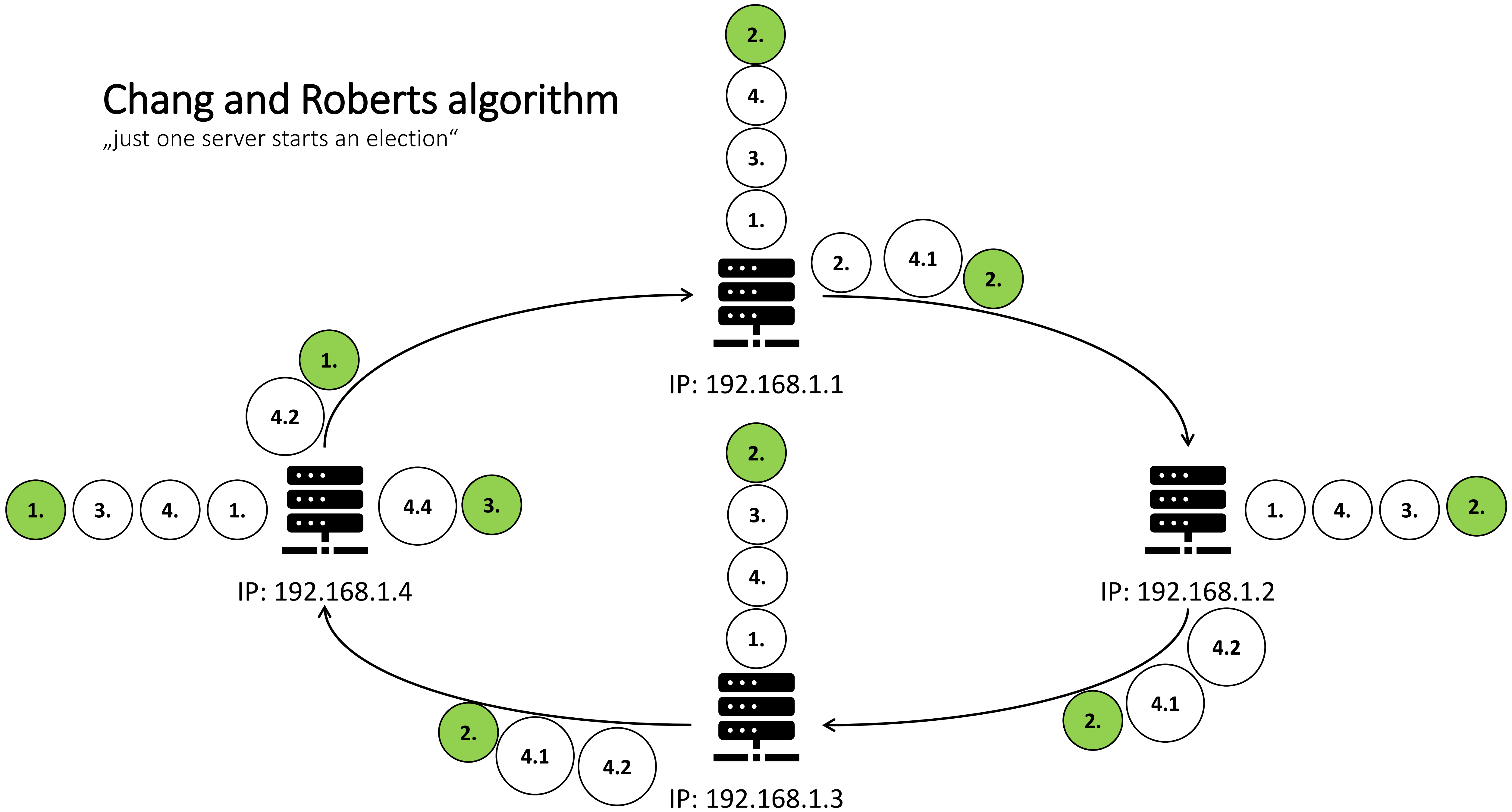
4.

Inform the front end of the current leader by leader message LE: LE(192.168.1.4)

Send LE to the frontend server

Chang and Roberts algorithm

„just one server starts an election“

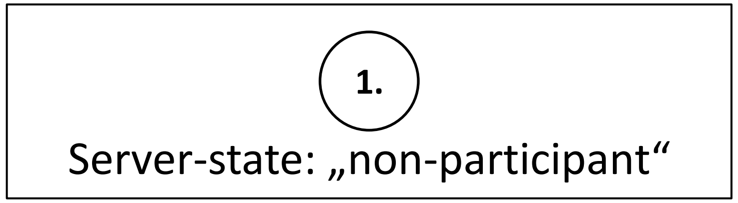


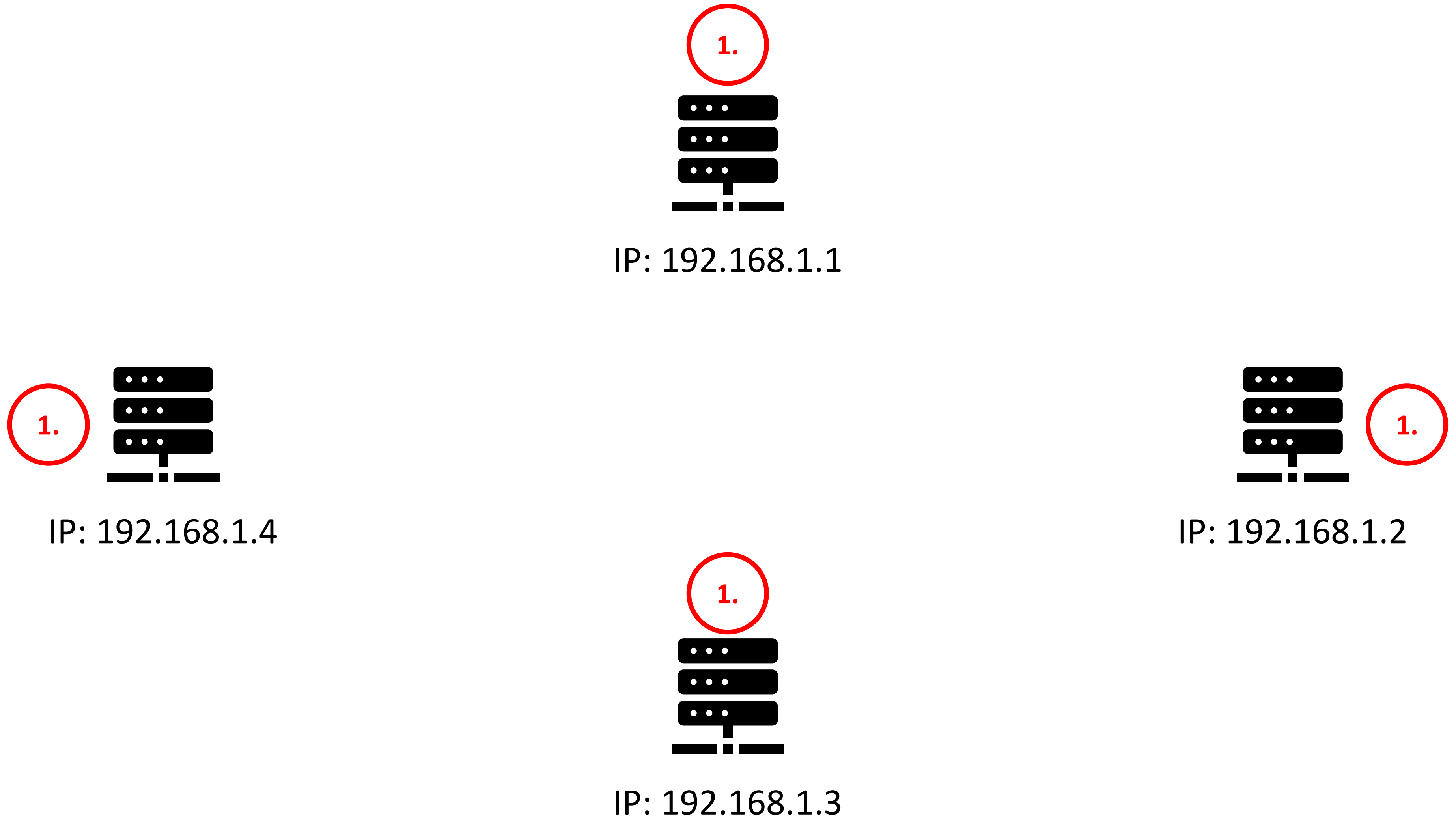
Chang and Roberts algorithm

Every server starts an election simultaneously

Chang and Roberts algorithm

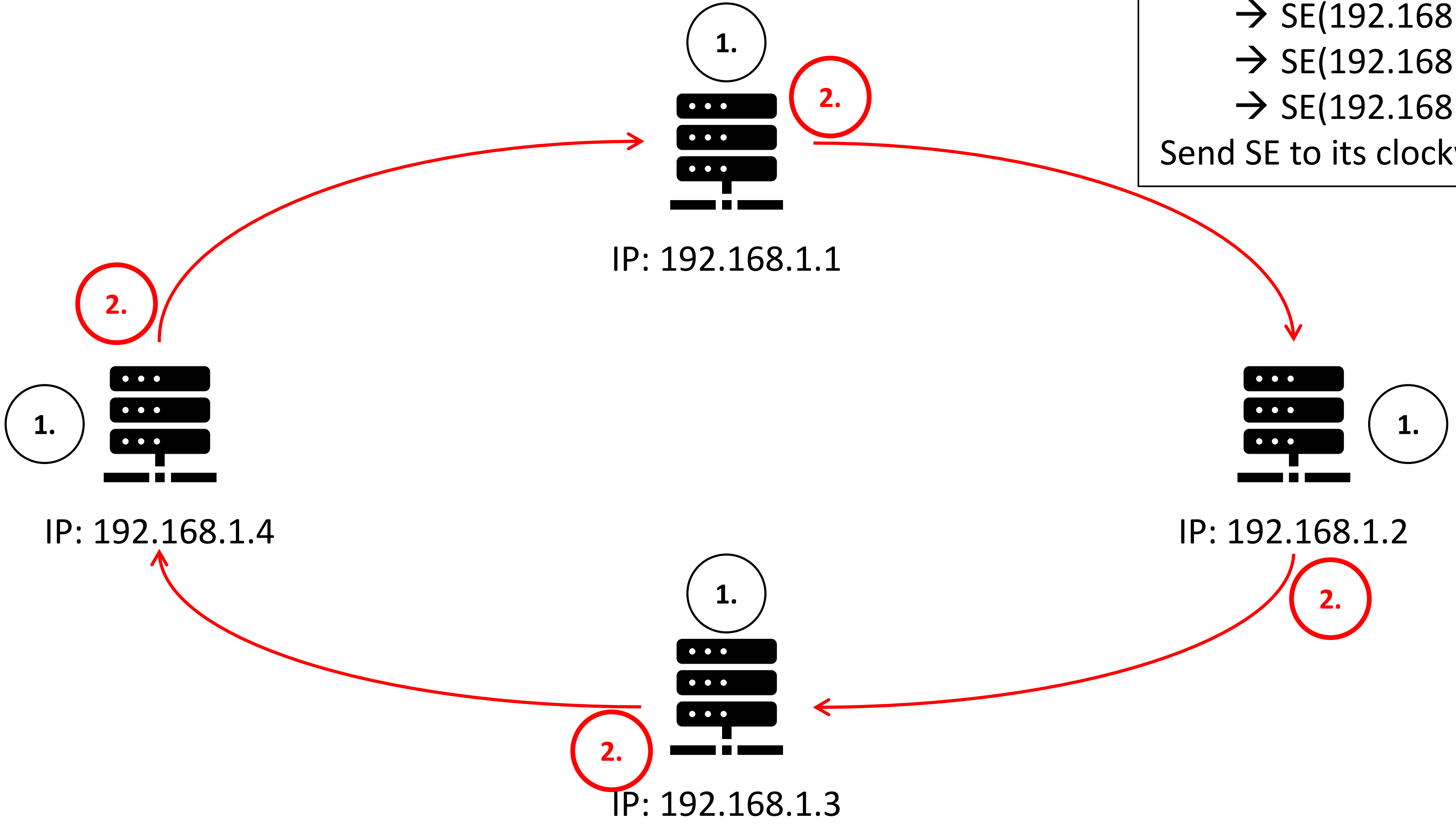
„Every server starts an election simultaneously“





Chang and Roberts algorithm

„Every server starts an election simultaneously“

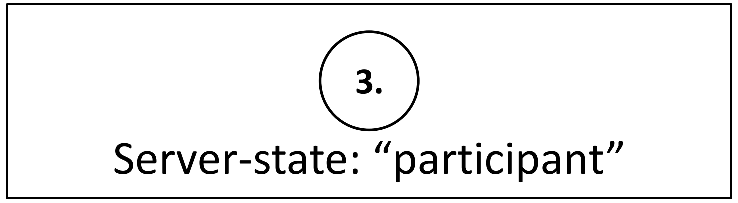


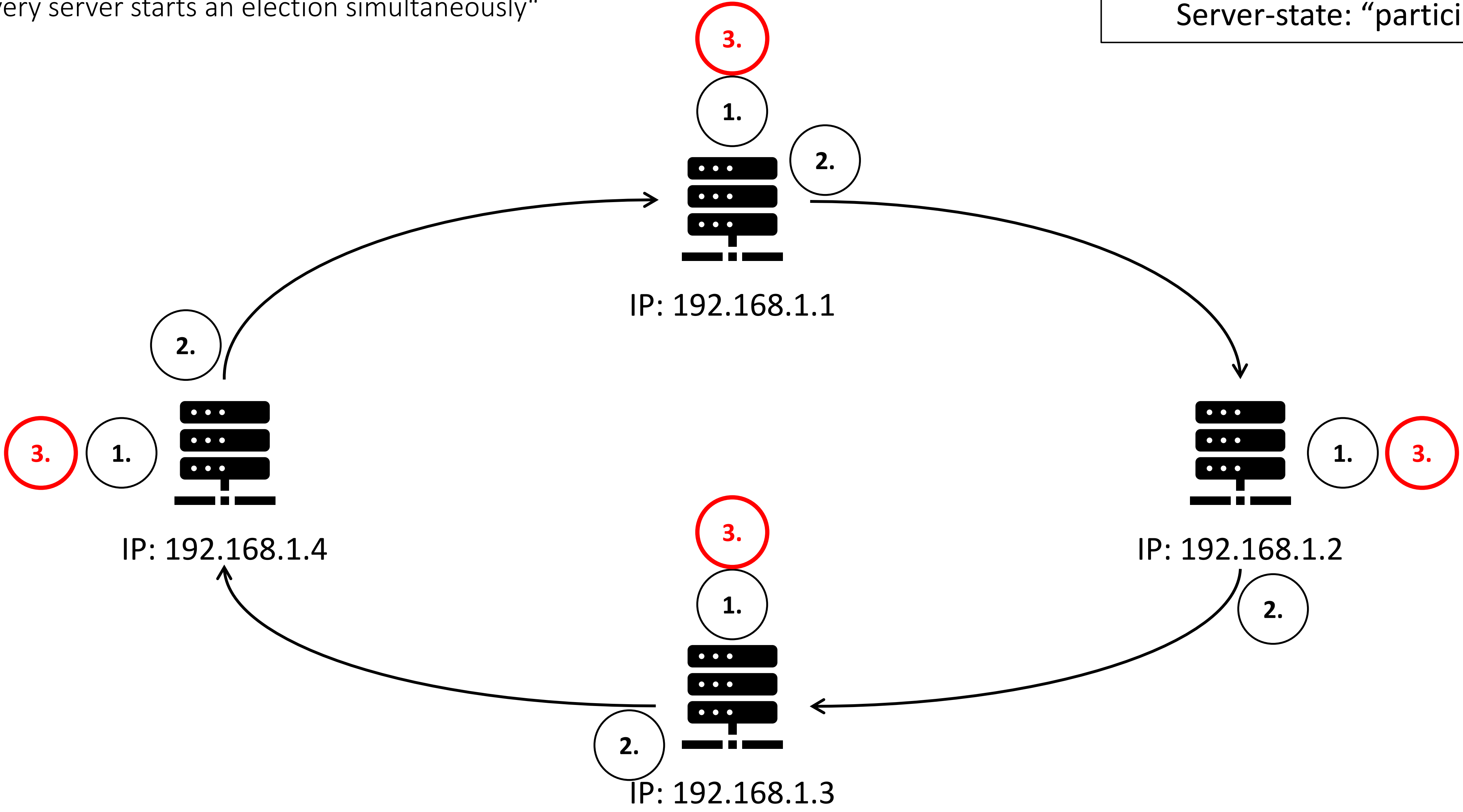
2.

Creation of election message SE:
SE(pid, isLeader)
→ SE(192.168.1.3, False)
→ SE(192.168.1.2, False)
→ SE(192.168.1.1, False)
→ SE(192.168.1.4, False)
Send SE to its clockwise neighbour

Chang and Roberts algorithm

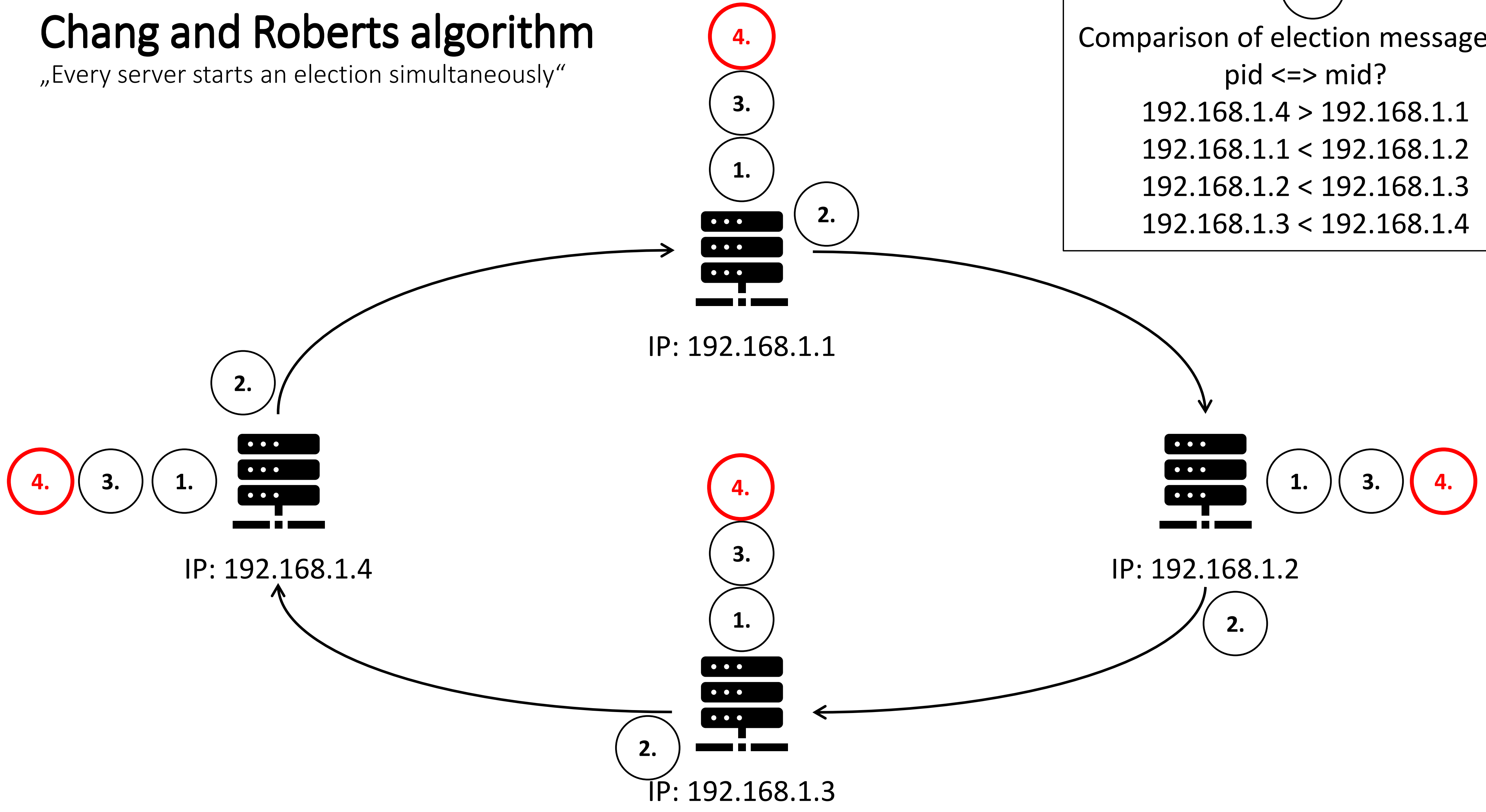
„Every server starts an election simultaneously“





Chang and Roberts algorithm

„Every server starts an election simultaneously“



4.

Comparison of election message SE:

pid <=> mid?

192.168.1.4 > 192.168.1.1

192.168.1.1 < 192.168.1.2

192.168.1.2 < 192.168.1.3

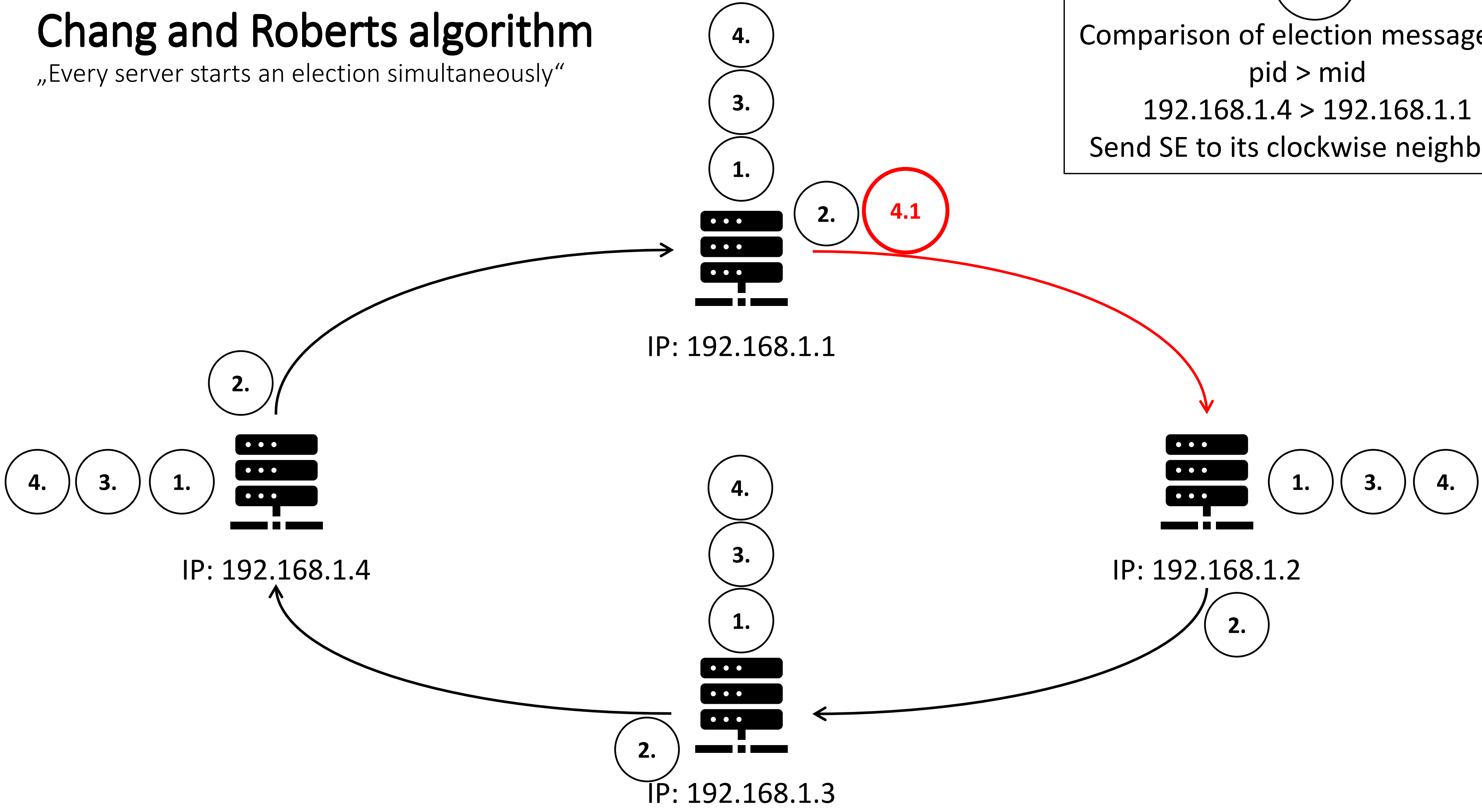
192.168.1.3 < 192.168.1.4

Chang and Roberts algorithm

„Every server starts an election simultaneously“

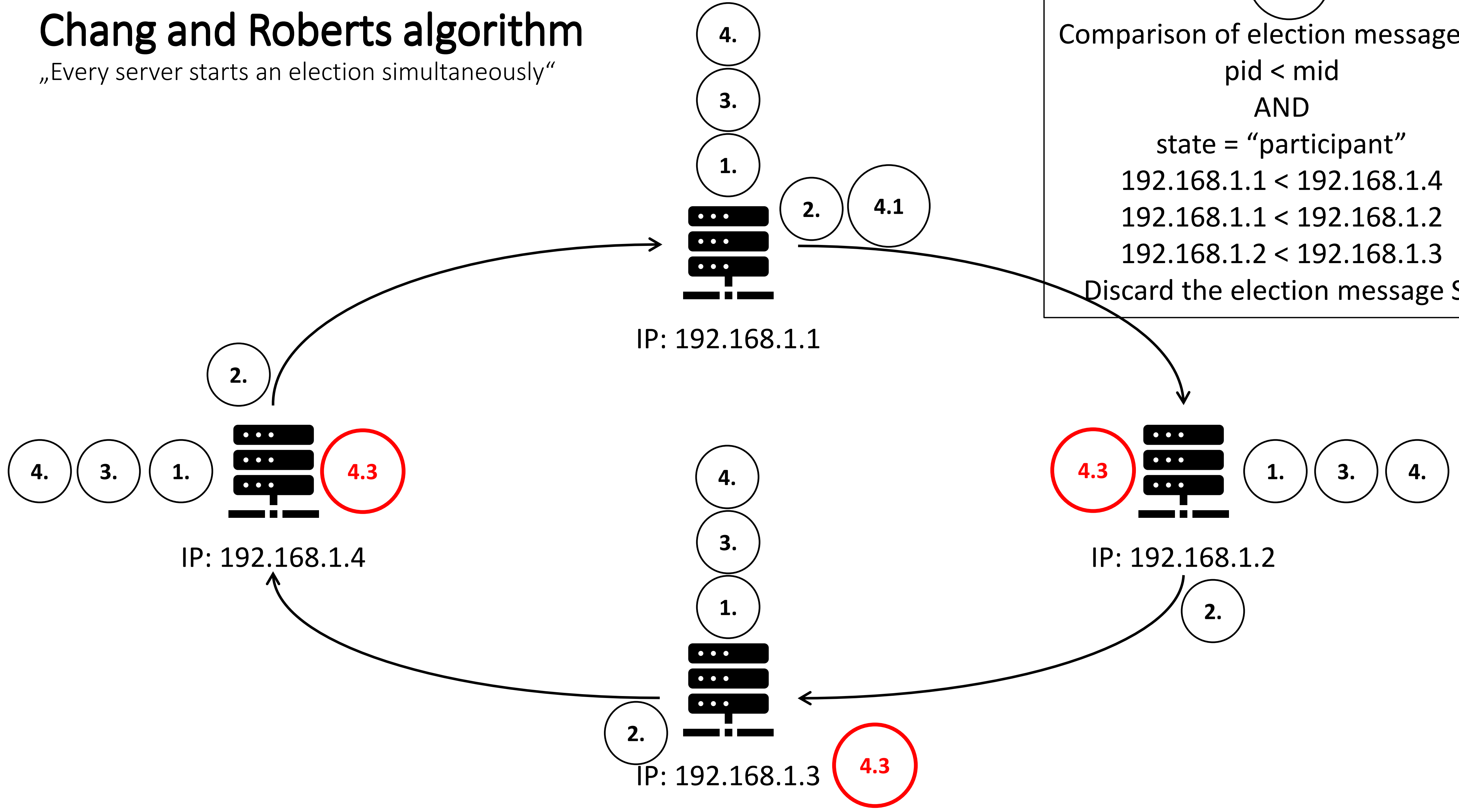
4.1

Comparison of election message SE:
pid > mid
192.168.1.4 > 192.168.1.1
Send SE to its clockwise neighbour



Chang and Roberts algorithm

„Every server starts an election simultaneously“



4.3

Comparison of election message SE:

pid < mid

AND

state = “participant”

192.168.1.1 < 192.168.1.4

192.168.1.1 < 192.168.1.2

192.168.1.2 < 192.168.1.3

Discard the election message SE

Chang and Roberts algorithm

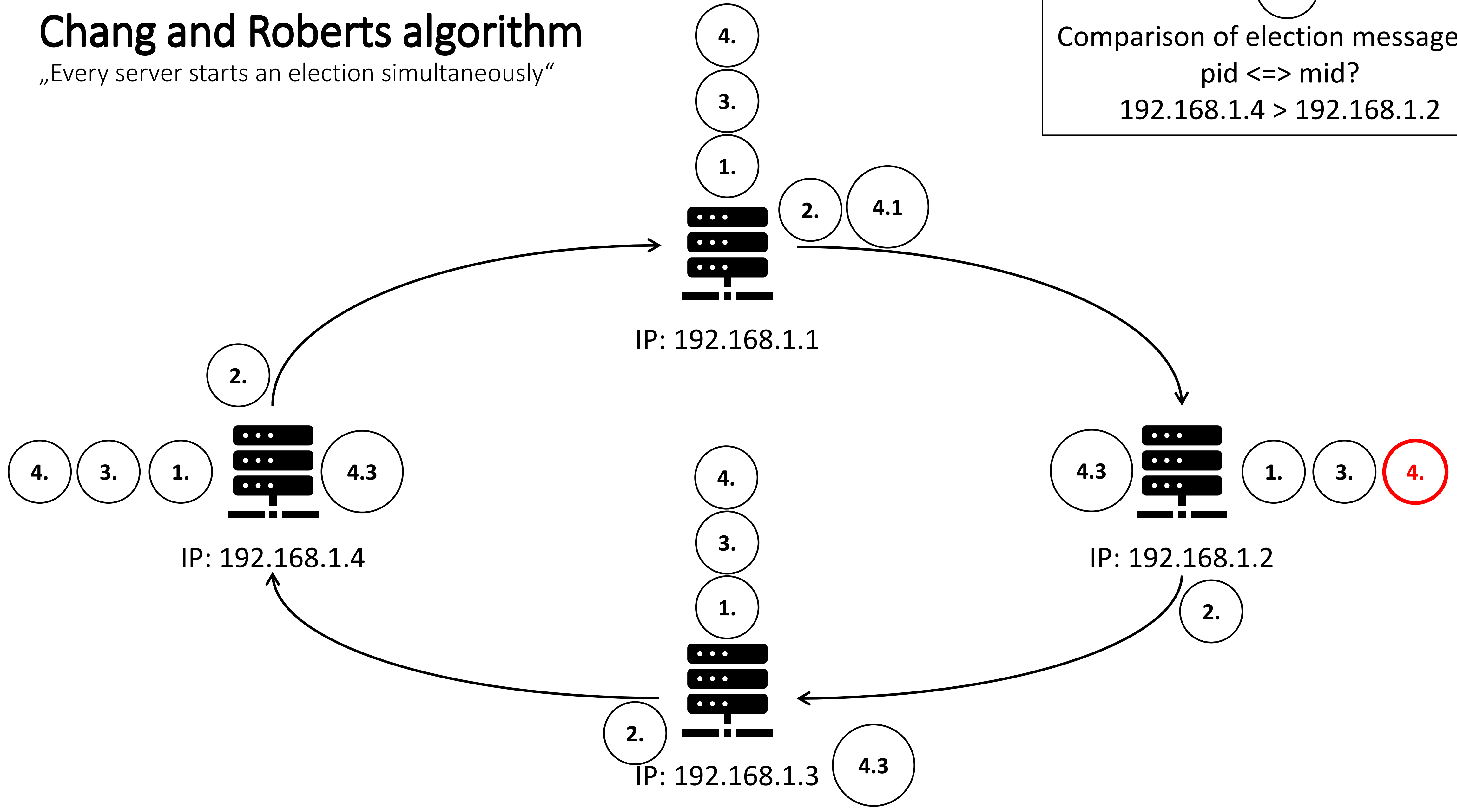
„Every server starts an election simultaneously“

4.

Comparison of election message SE:

pid <=> mid?

192.168.1.4 > 192.168.1.2



Chang and Roberts algorithm

„Every server starts an election simultaneously“

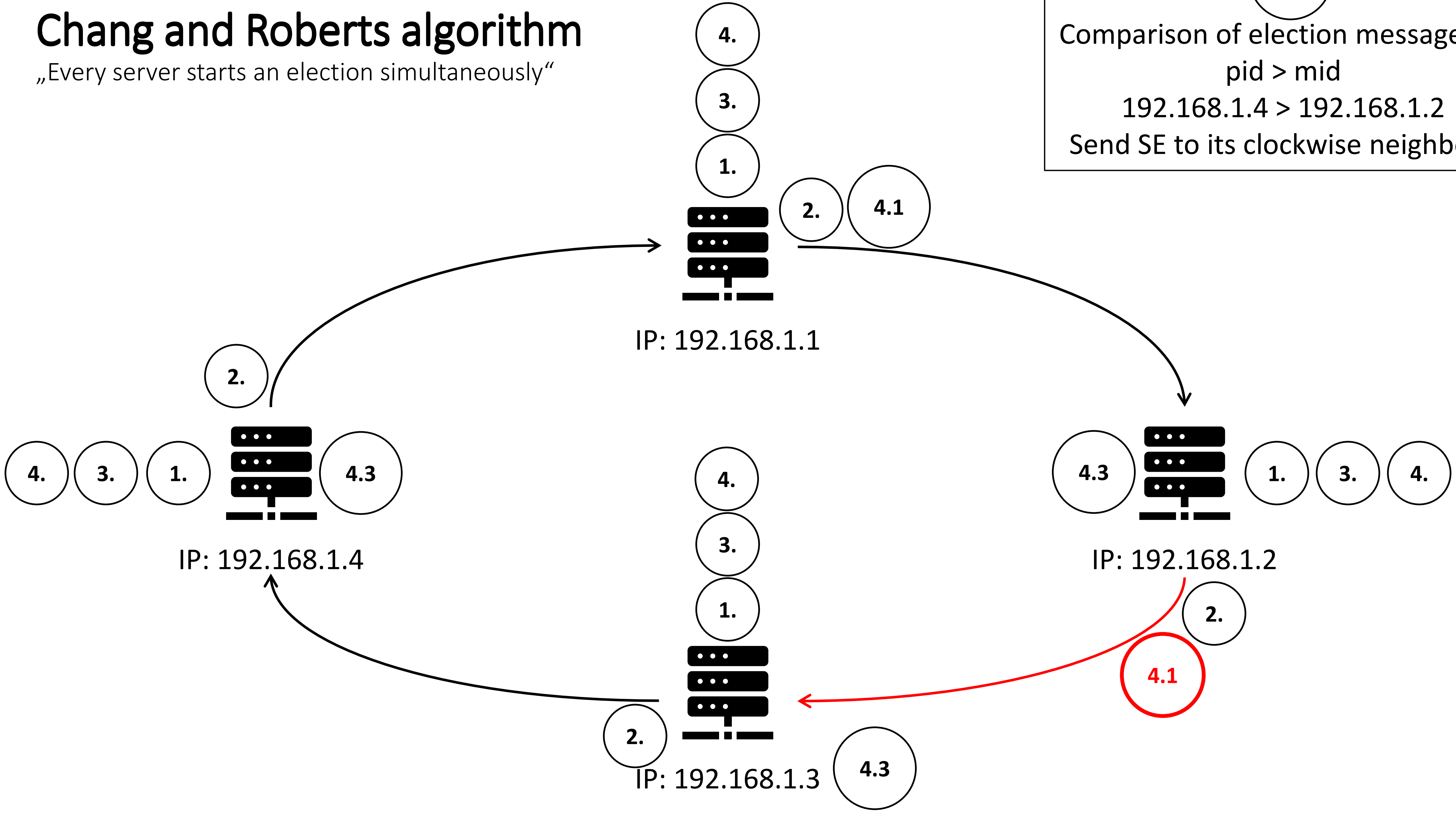
4.1

Comparison of election message SE:

pid > mid

192.168.1.4 > 192.168.1.2

Send SE to its clockwise neighbour



Chang and Roberts algorithm

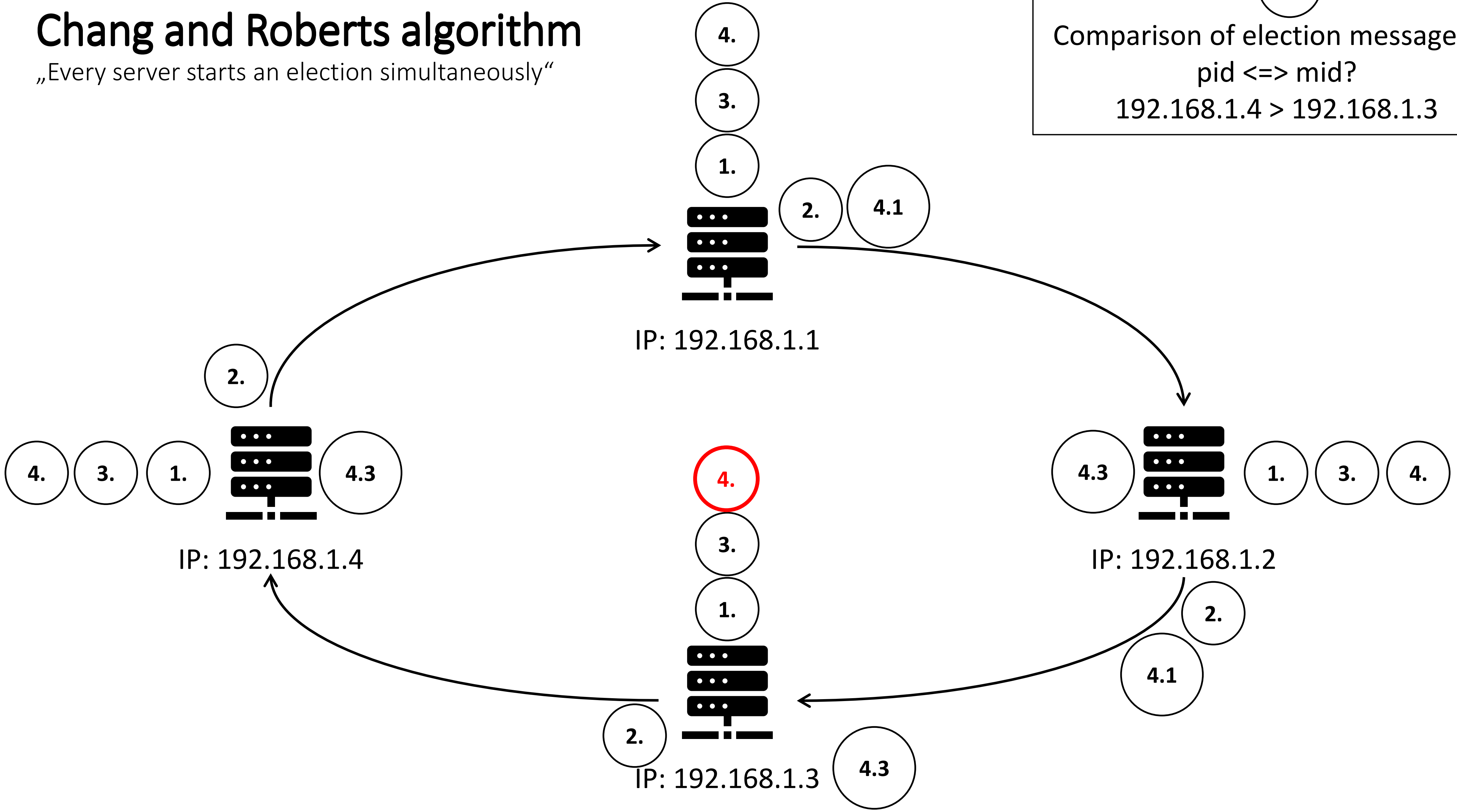
„Every server starts an election simultaneously“

4\.

Comparison of election message SE:

pid <=> mid?

192.168.1.4 > 192.168.1.3



Chang and Roberts algorithm

„Every server starts an election simultaneously“

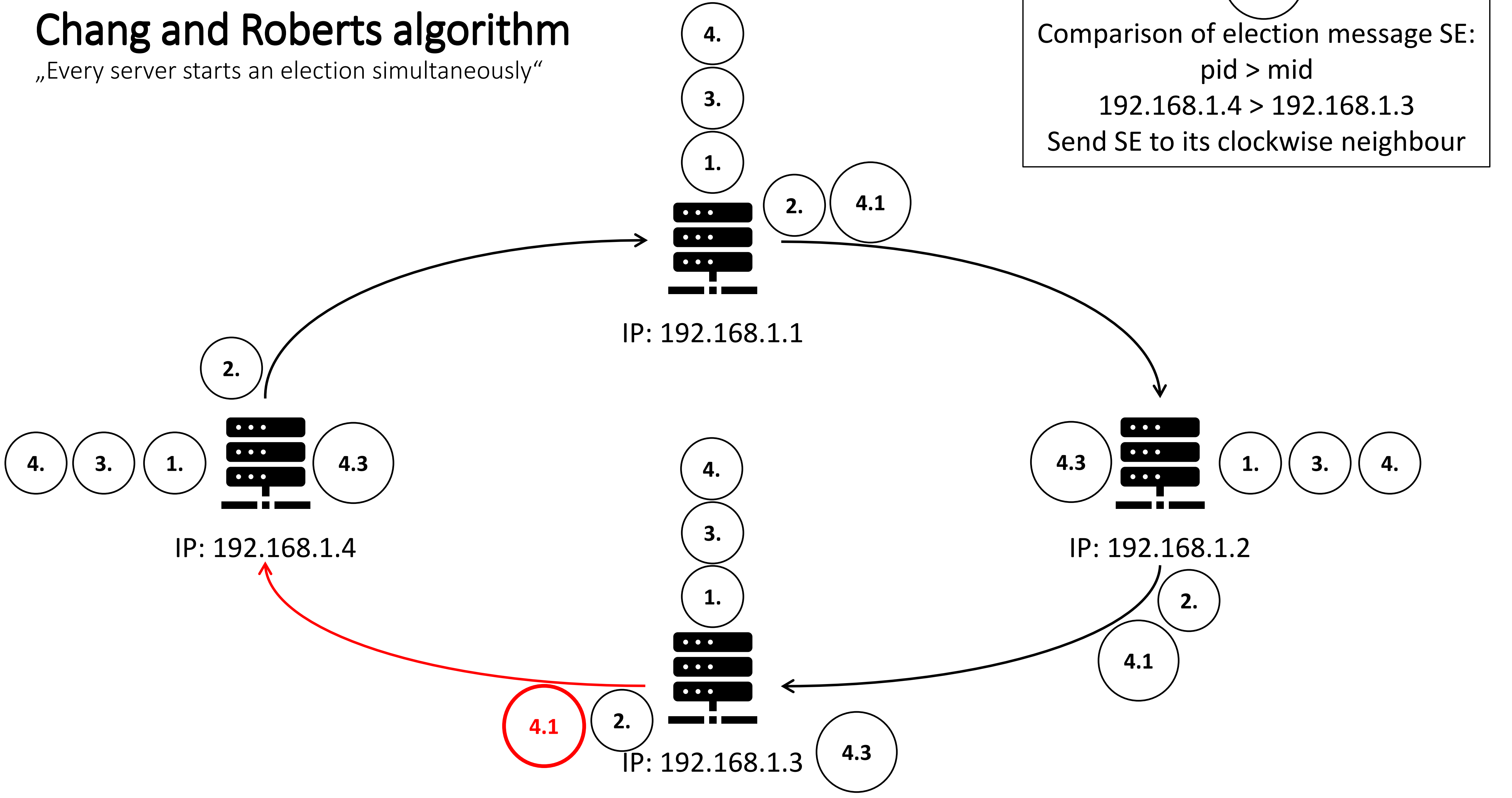
4.1

Comparison of election message SE:

pid > mid

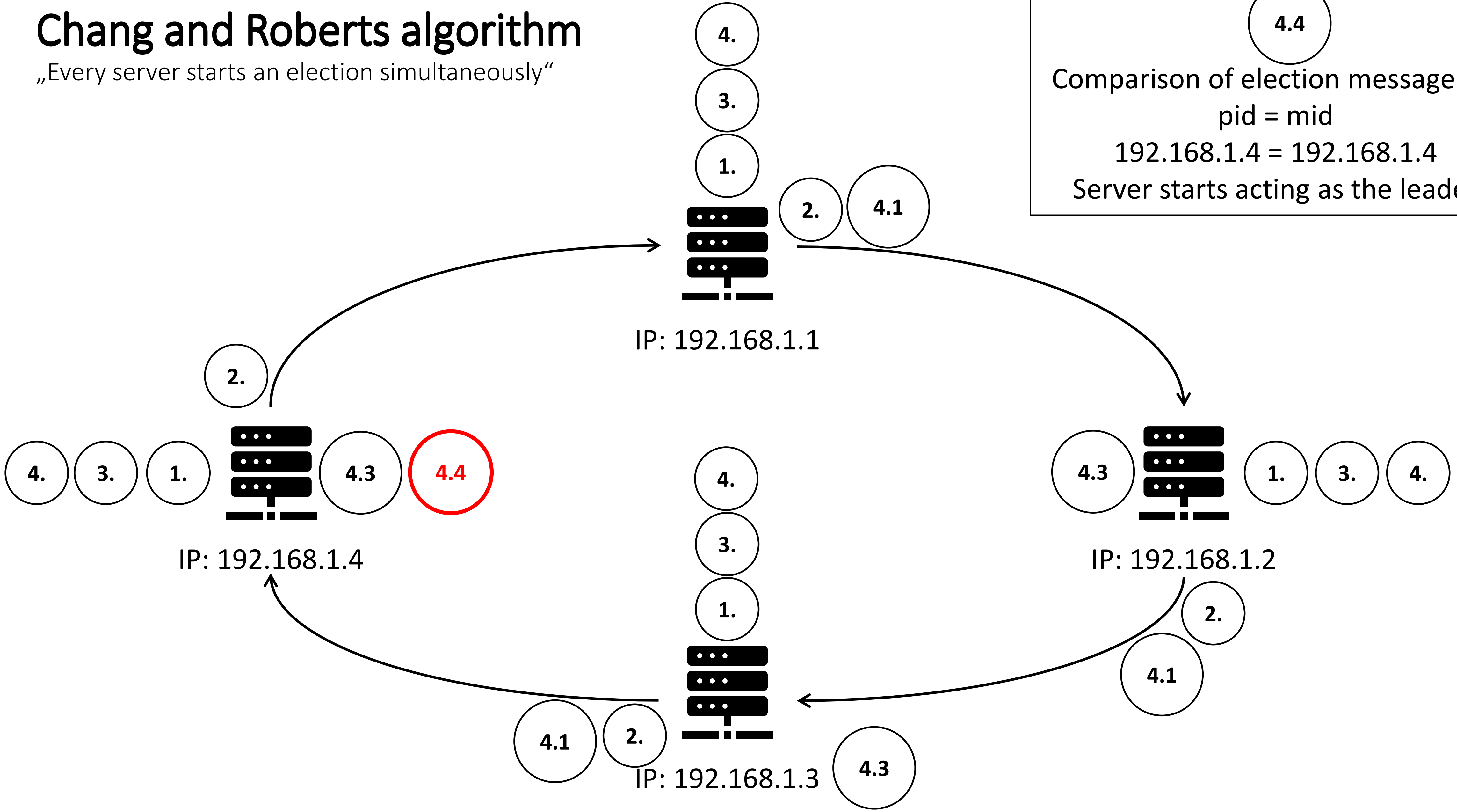
192.168.1.4 > 192.168.1.3

Send SE to its clockwise neighbour



Chang and Roberts algorithm

„Every server starts an election simultaneously“



4.4

Comparison of election message SE:

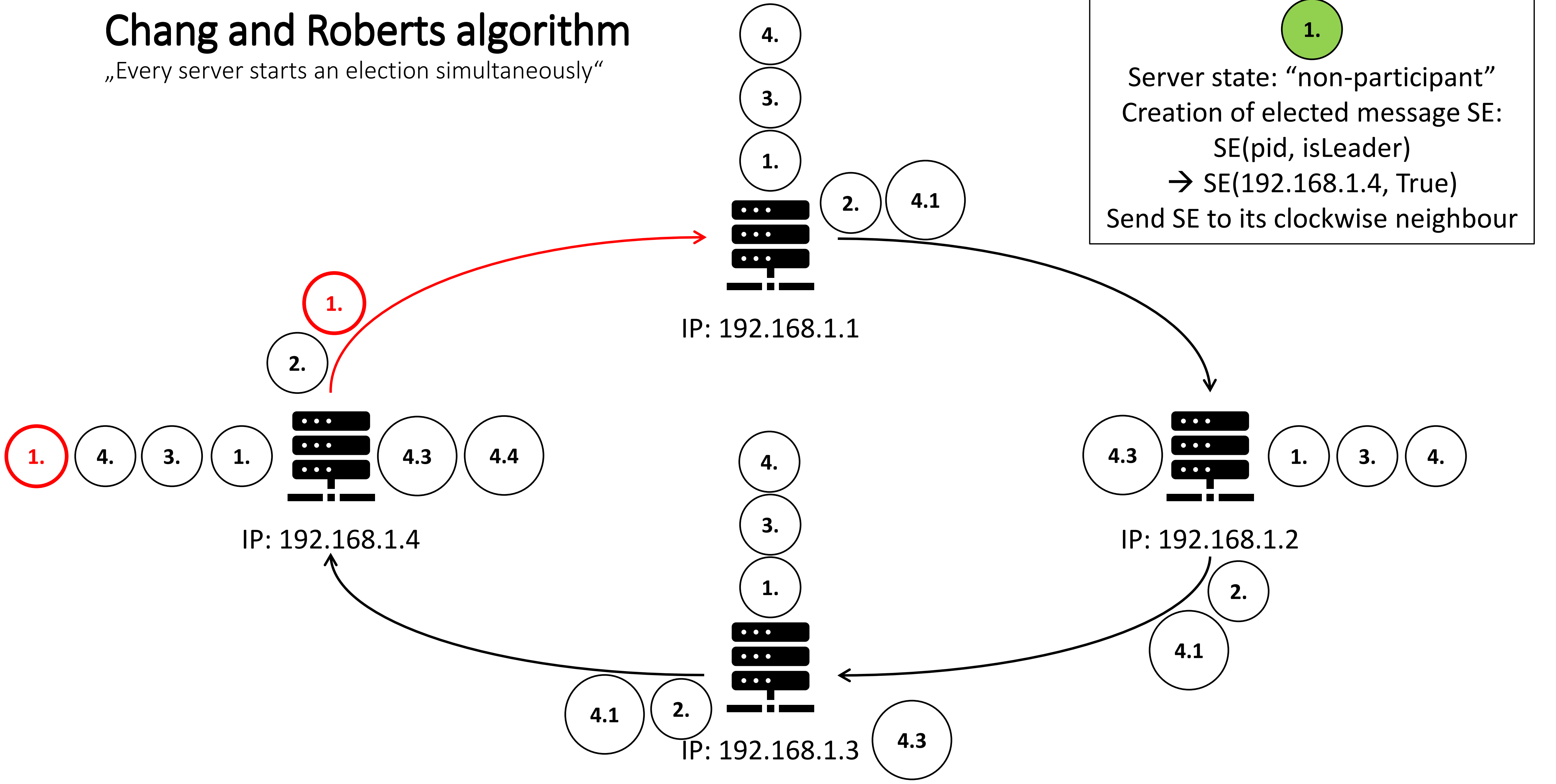
pid = mid

192.168.1.4 = 192.168.1.4

Server starts acting as the leader

Chang and Roberts algorithm

„Every server starts an election simultaneously“



1.

Server state: “non-participant”

Creation of elected message SE:

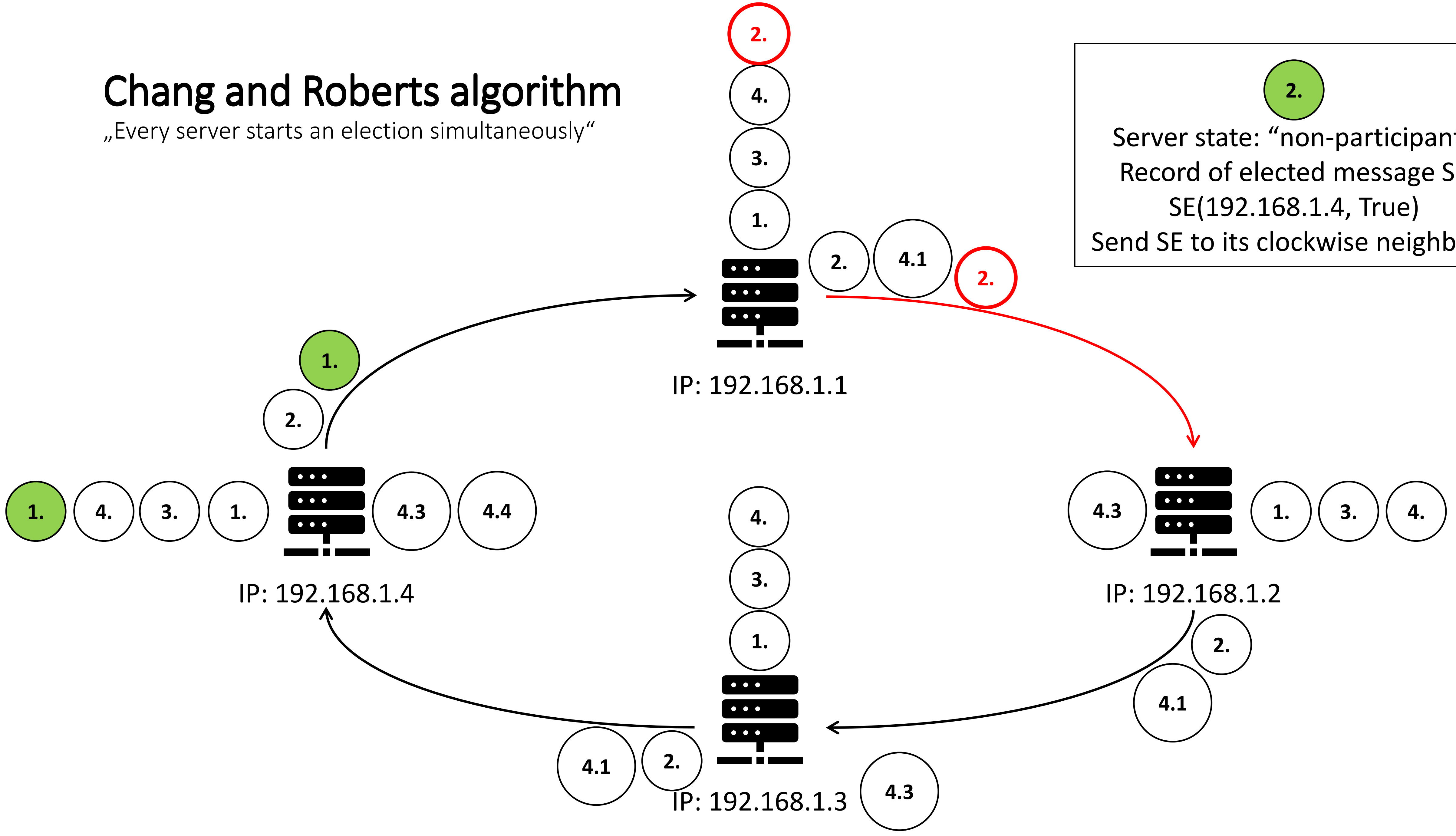
SE(pid, isLeader)

→ SE(192.168.1.4, True)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„Every server starts an election simultaneously“



2.

Server state: “non-participant”

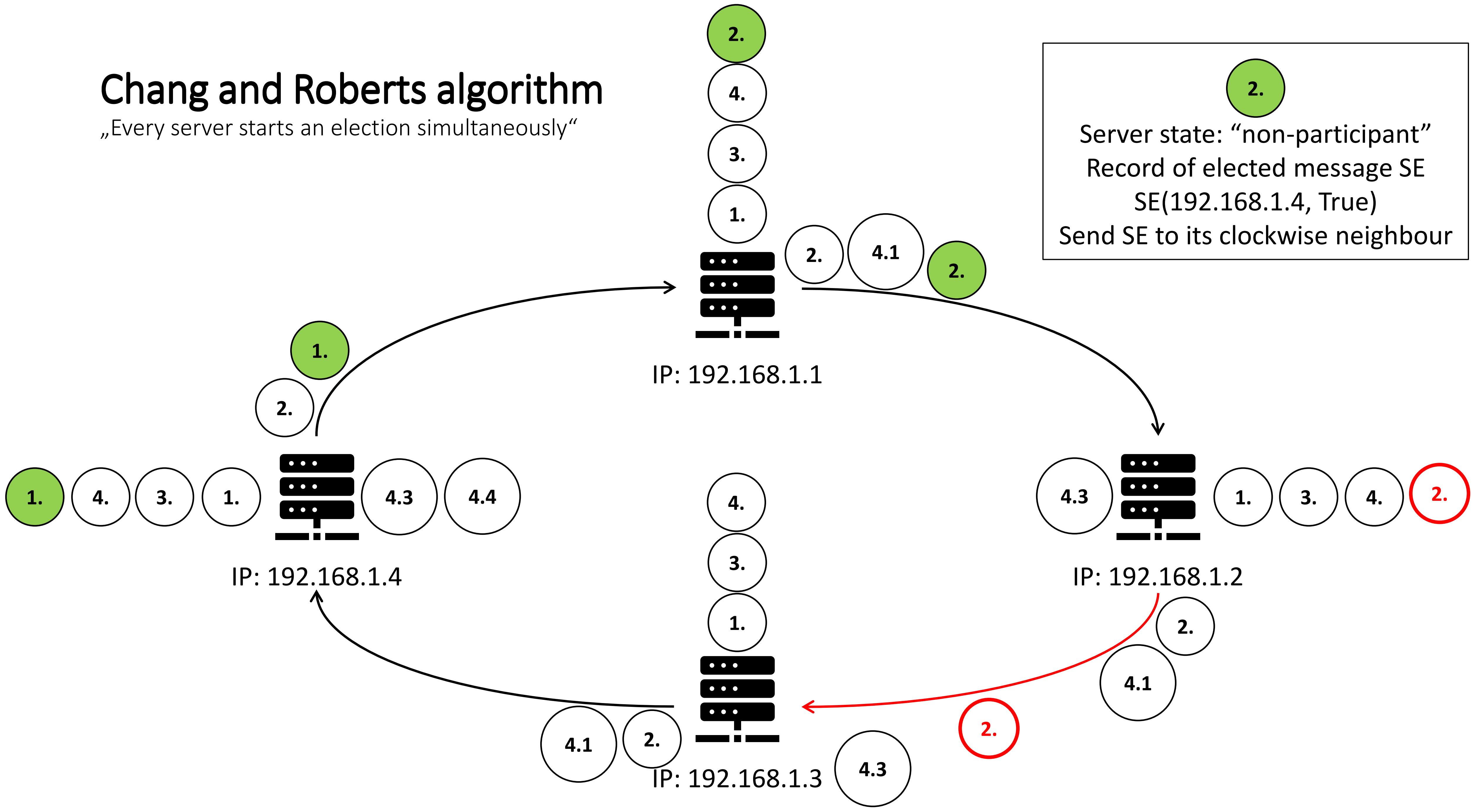
Record of elected message SE

SE(192.168.1.4, True)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„Every server starts an election simultaneously“



2.

Server state: “non-participant”

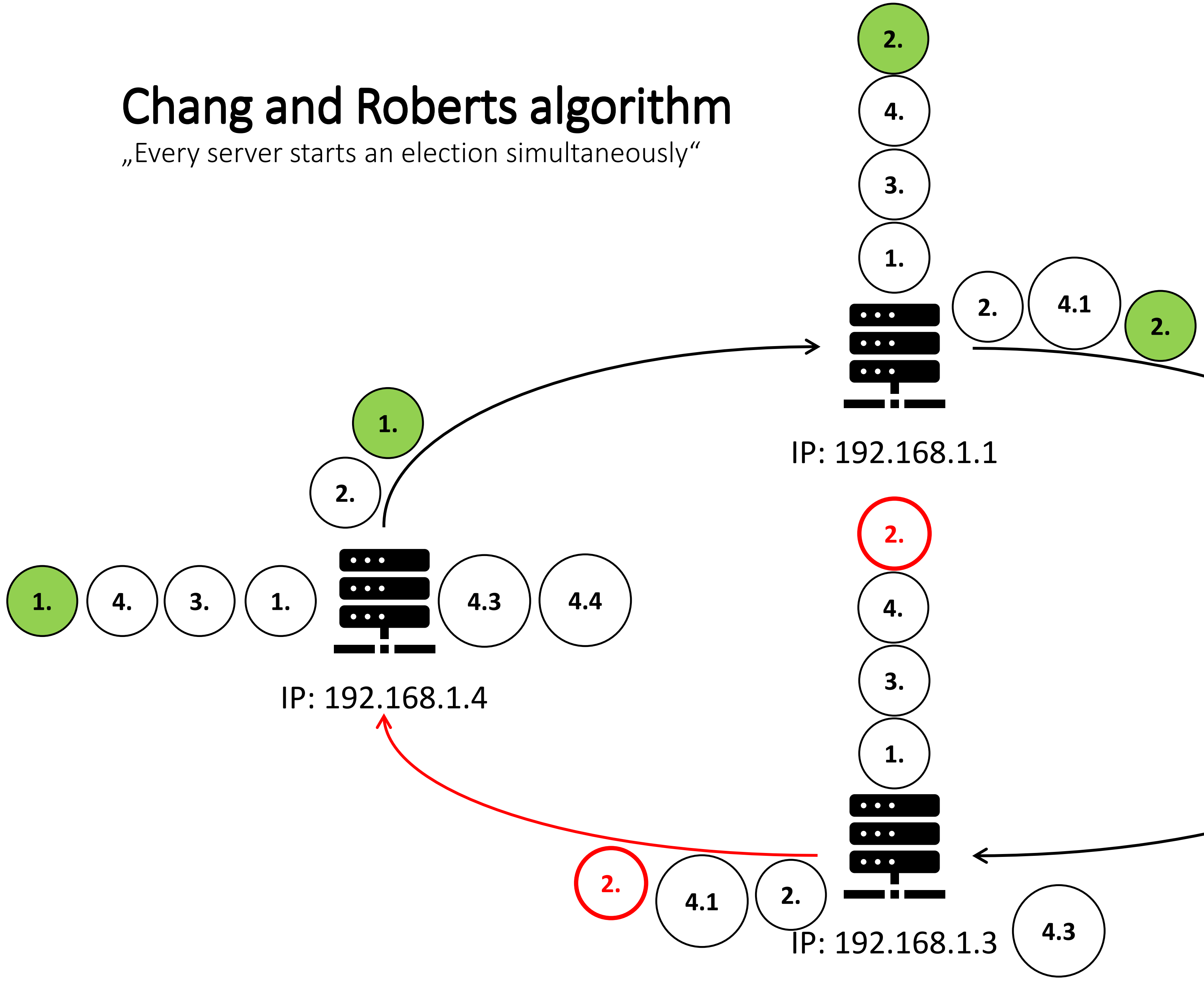
Record of elected message SE

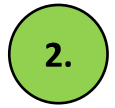
SE(192.168.1.4, True)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„Every server starts an election simultaneously“





Server state: “non-participant”

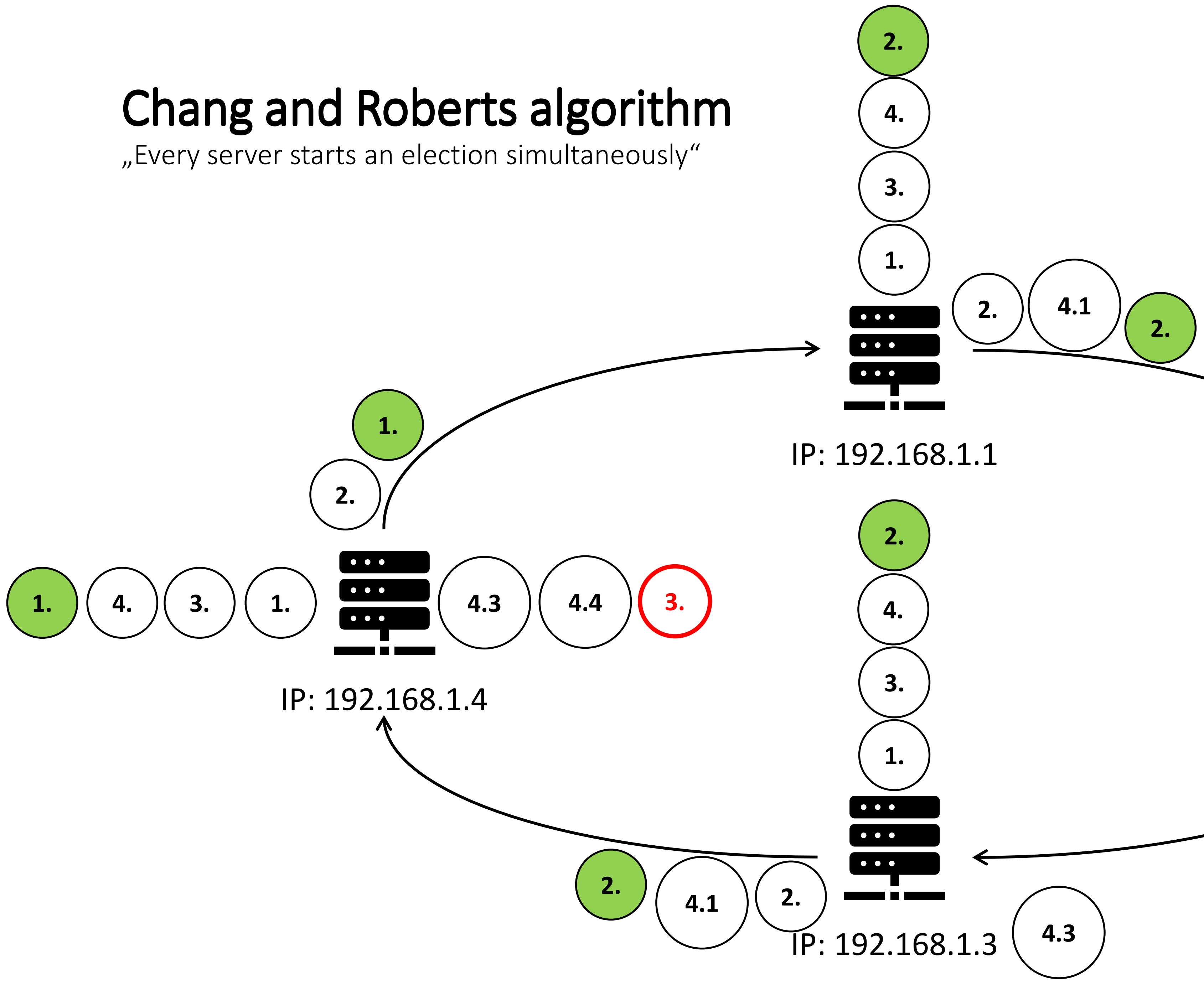
Record of elected message SE

SE(192.168.1.4, True)

Send SE to its clockwise neighbour

Chang and Roberts algorithm

„Every server starts an election simultaneously“

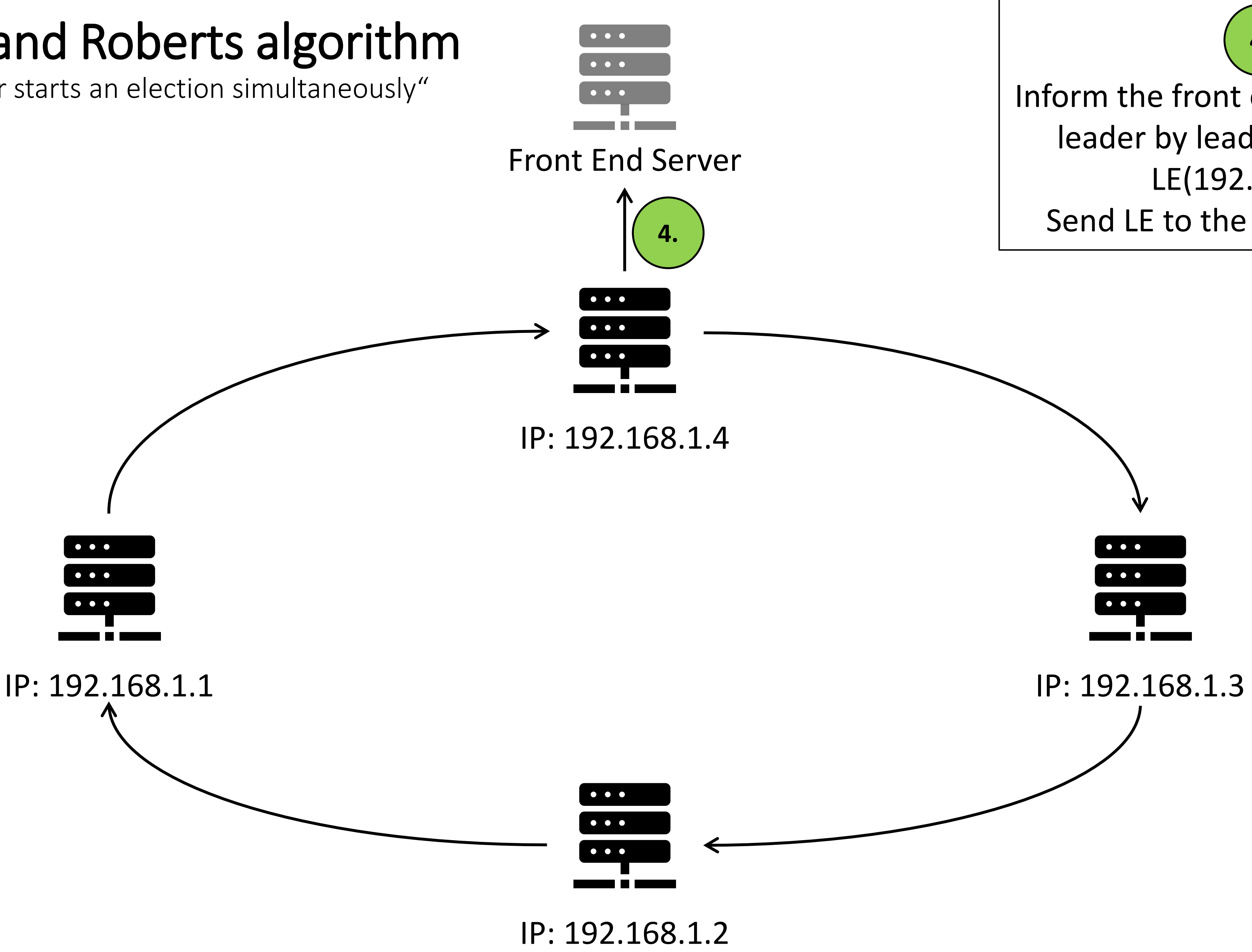


3.

Leader discards the message
→ Election is over

Chang and Roberts algorithm

„Every server starts an election simultaneously“

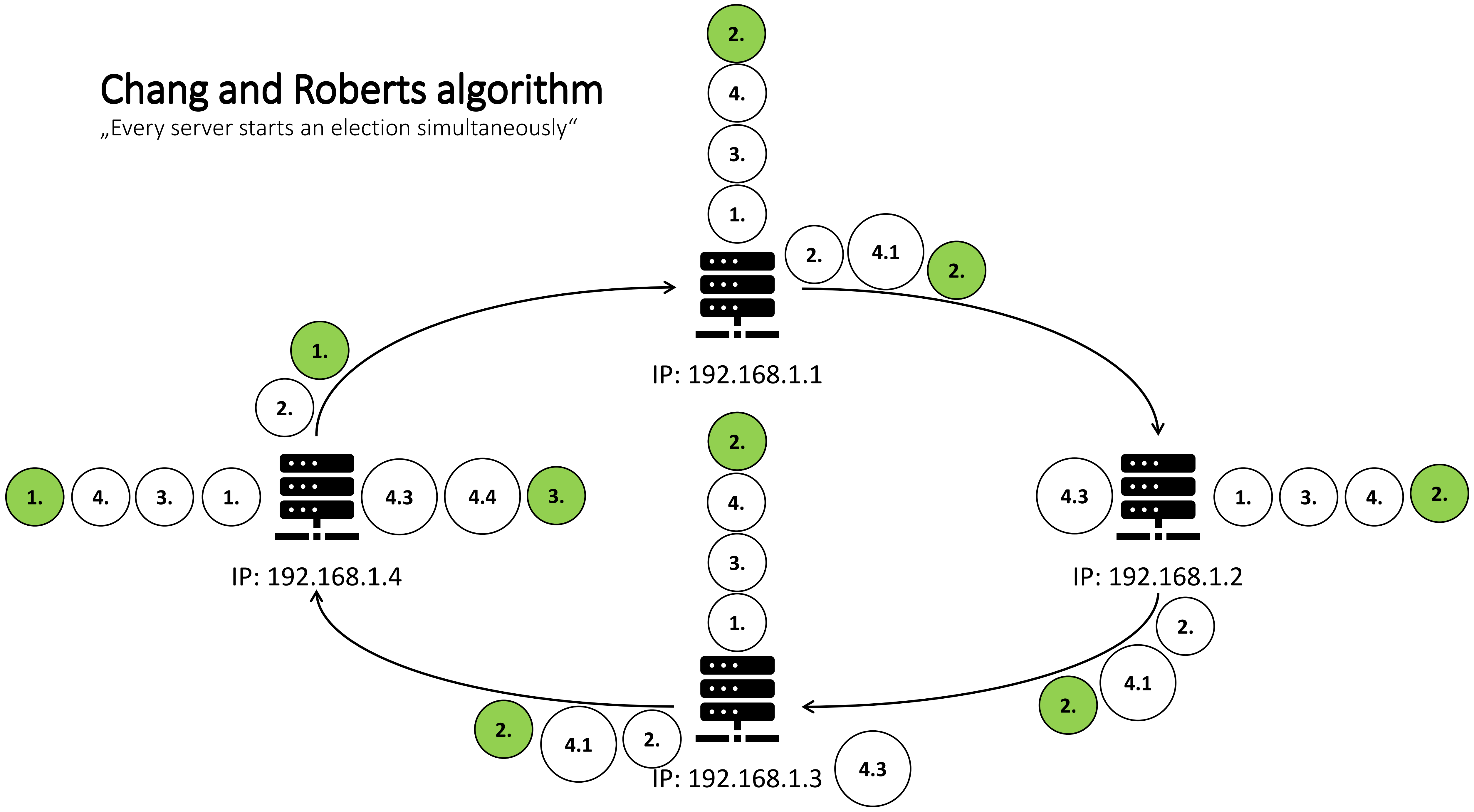


4.

Inform the front end of the current leader by leader message LE:
LE(192.168.1.4)
Send LE to the frontend server

Chang and Roberts algorithm

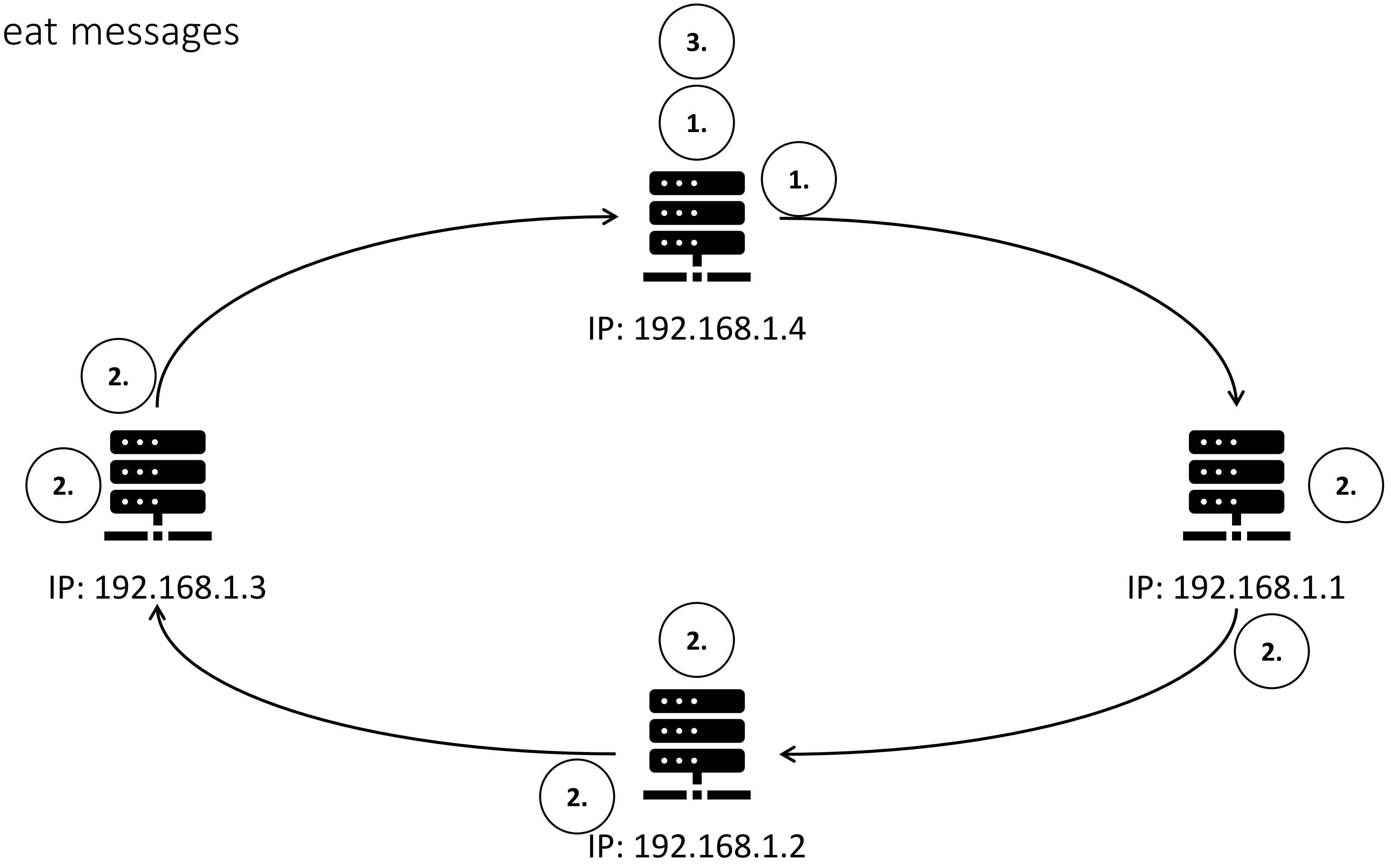
„Every server starts an election simultaneously“



Failure detector

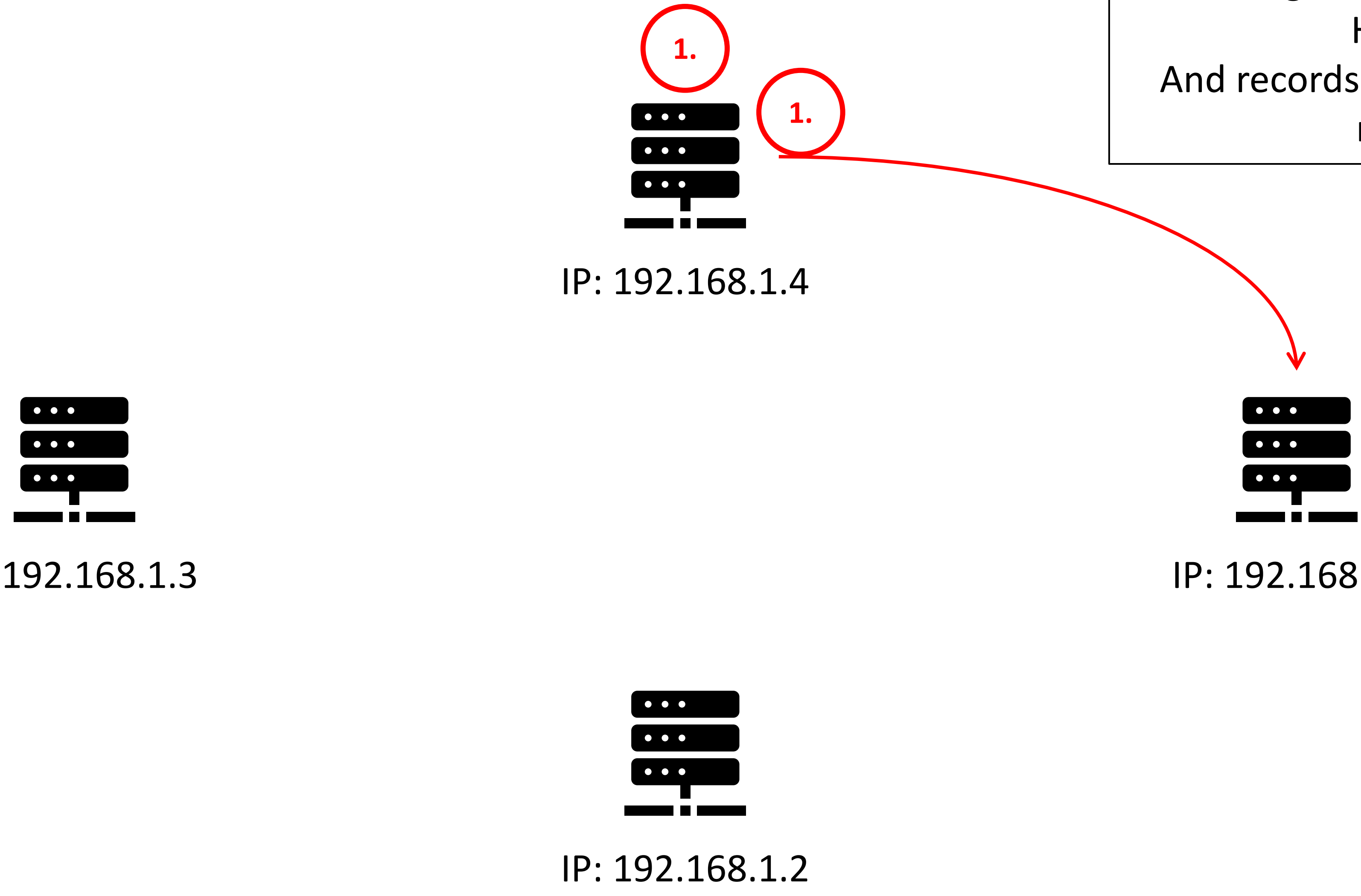
Heartbeat messages

Failure detector
Heartbeat messages



Failure detector

Heartbeat messages



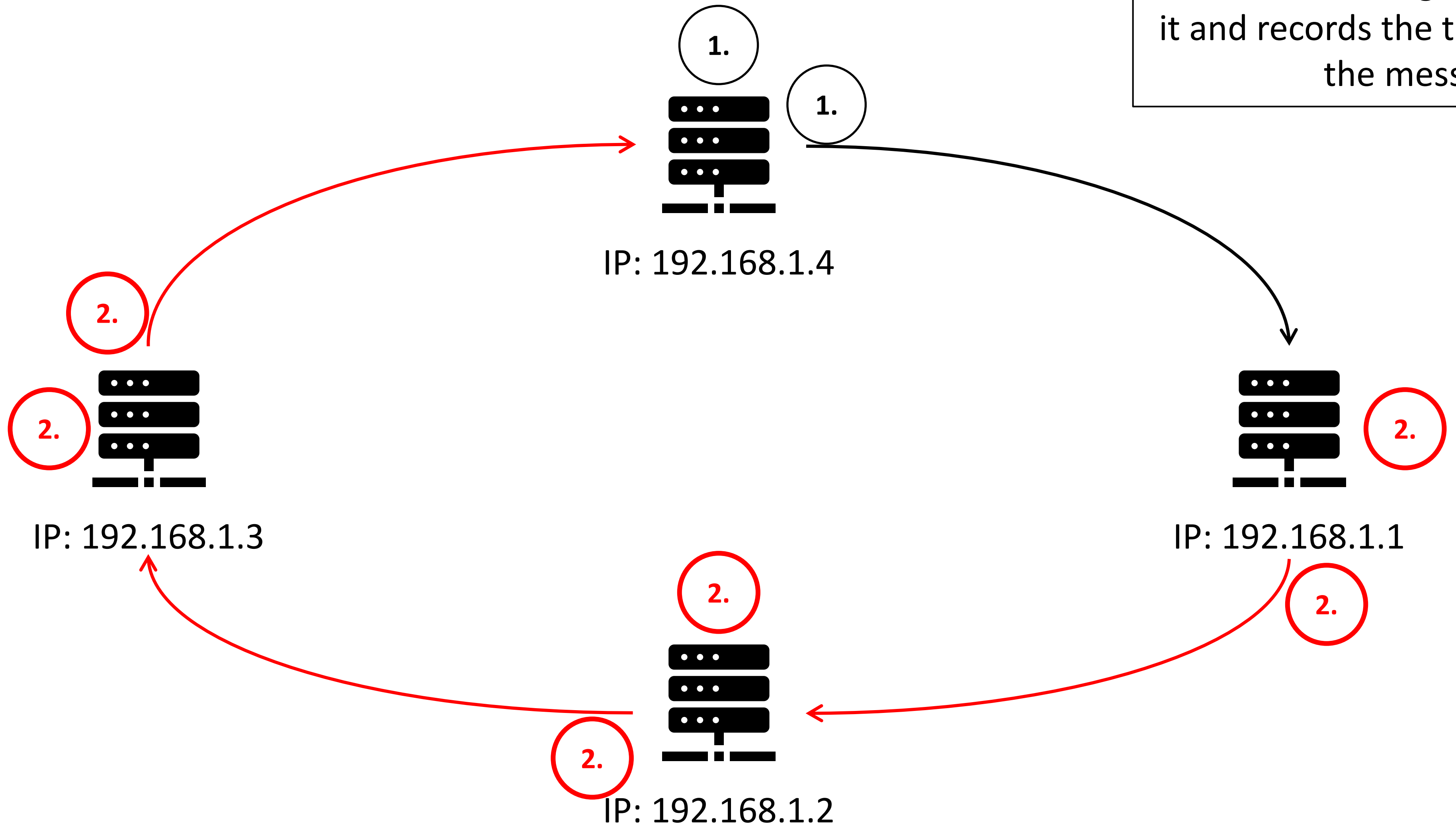
1.

Primary Server sends the heartbeat message HB(heartbeat GUID): HB(1234)

And records the time it send the message

Failure detector

Heartbeat messages

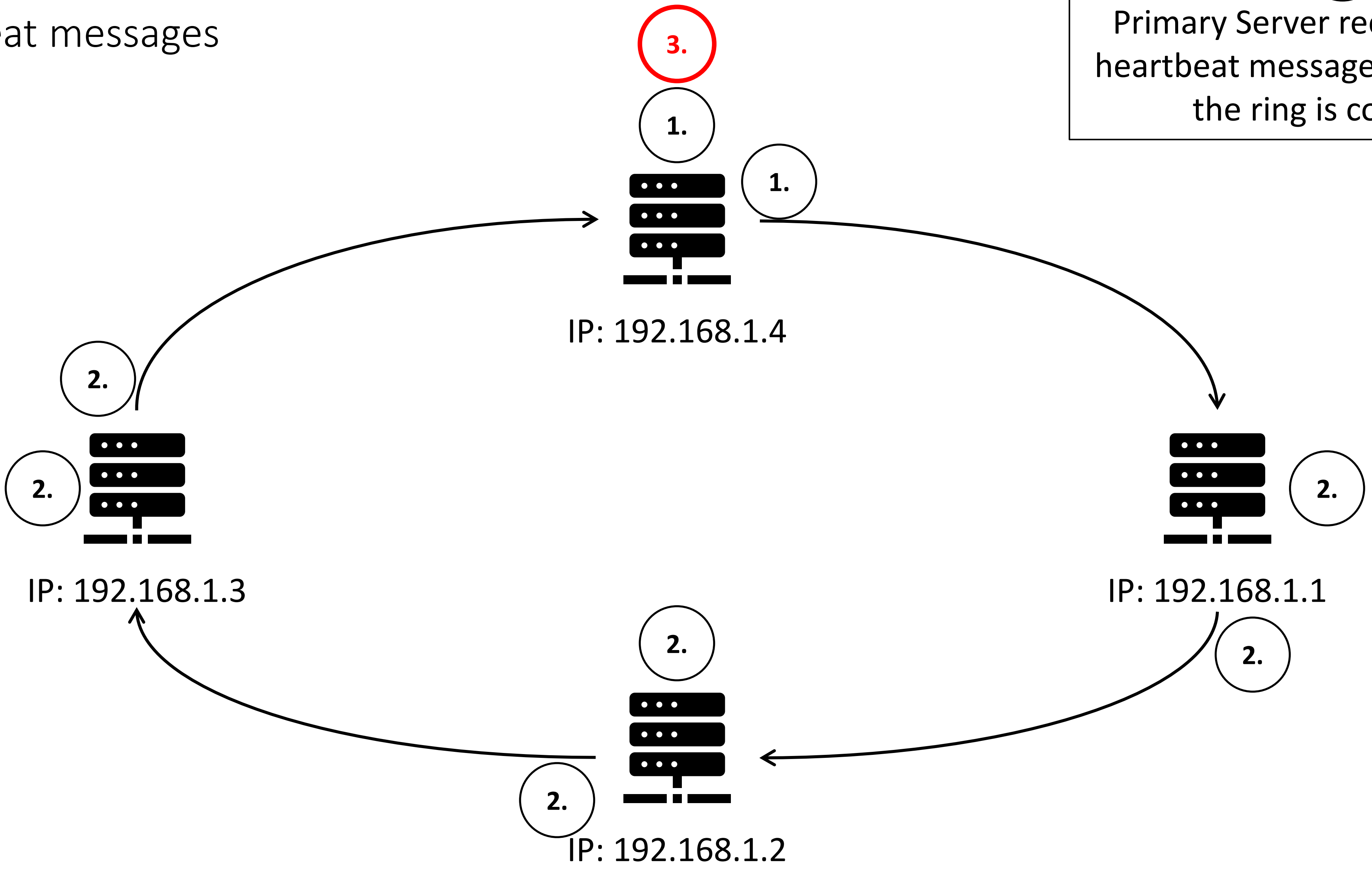


2.

Backup Server receives the heartbeat message(HB), forwards it and records the time it received the message

Failure detector

Heartbeat messages

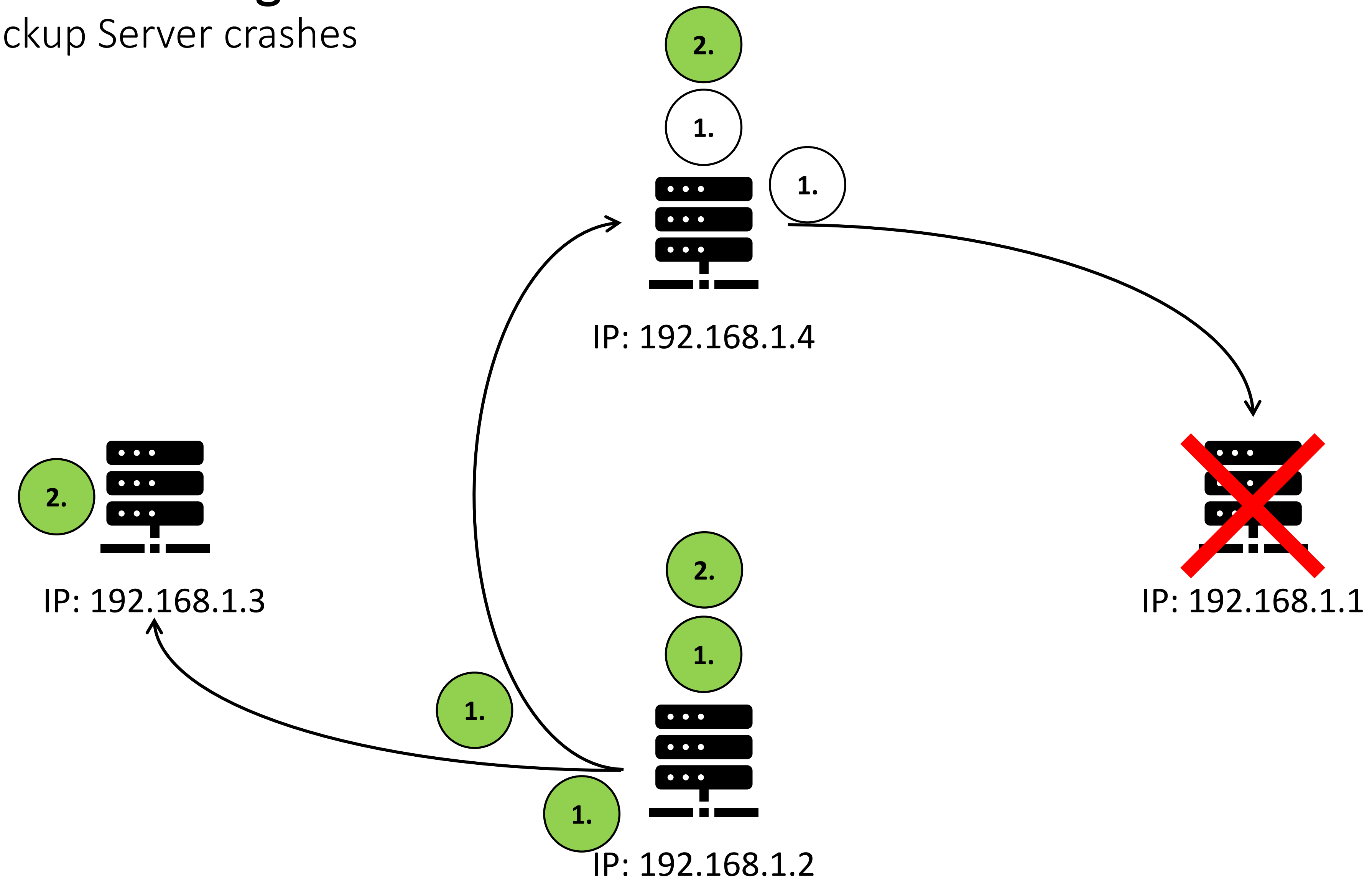


3.

Primary Server receives its own heartbeat message it records that the ring is complete.

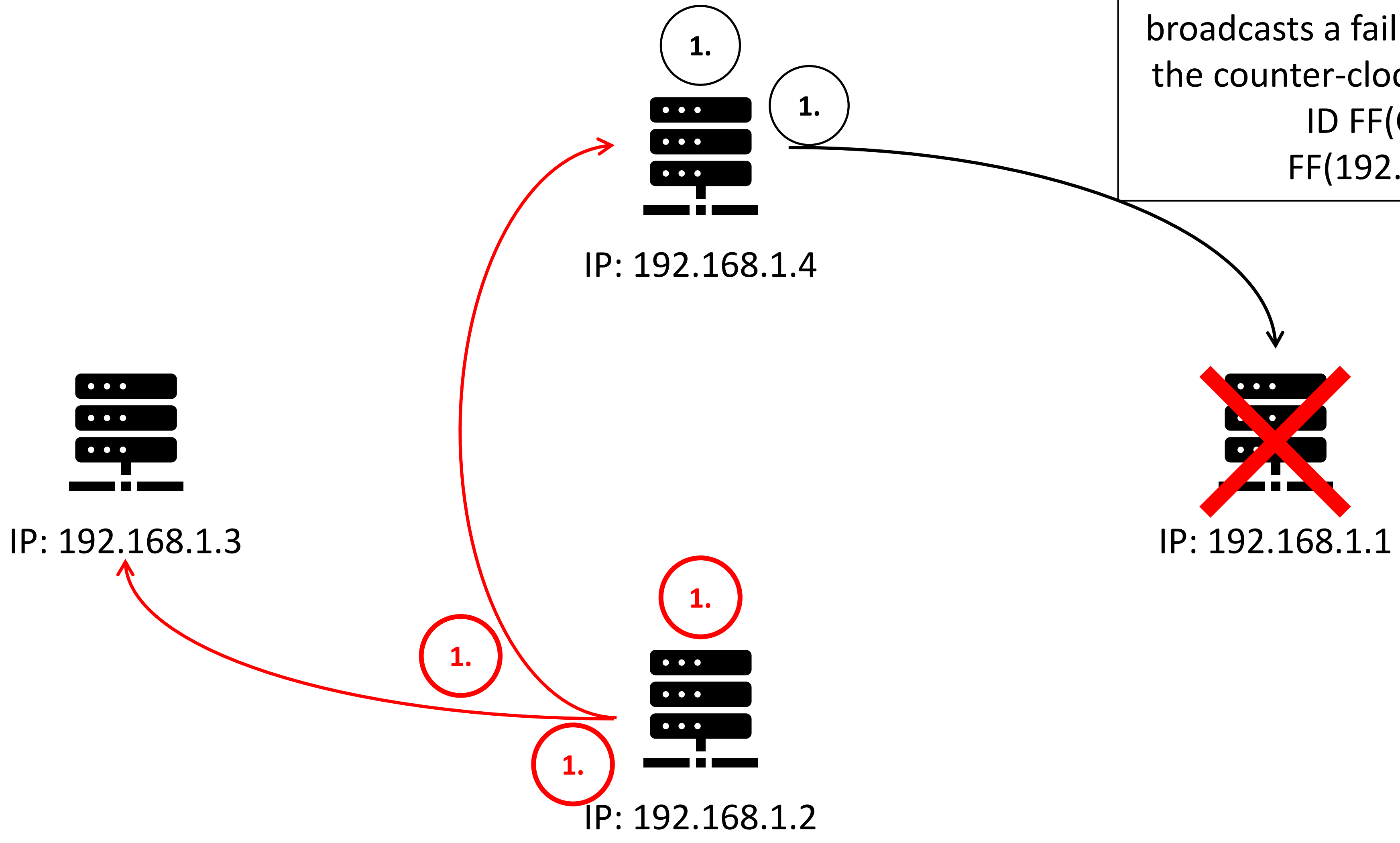
Heartbeat messages

One Backup Server crashes



Heartbeat messages

One Backup Server crashes

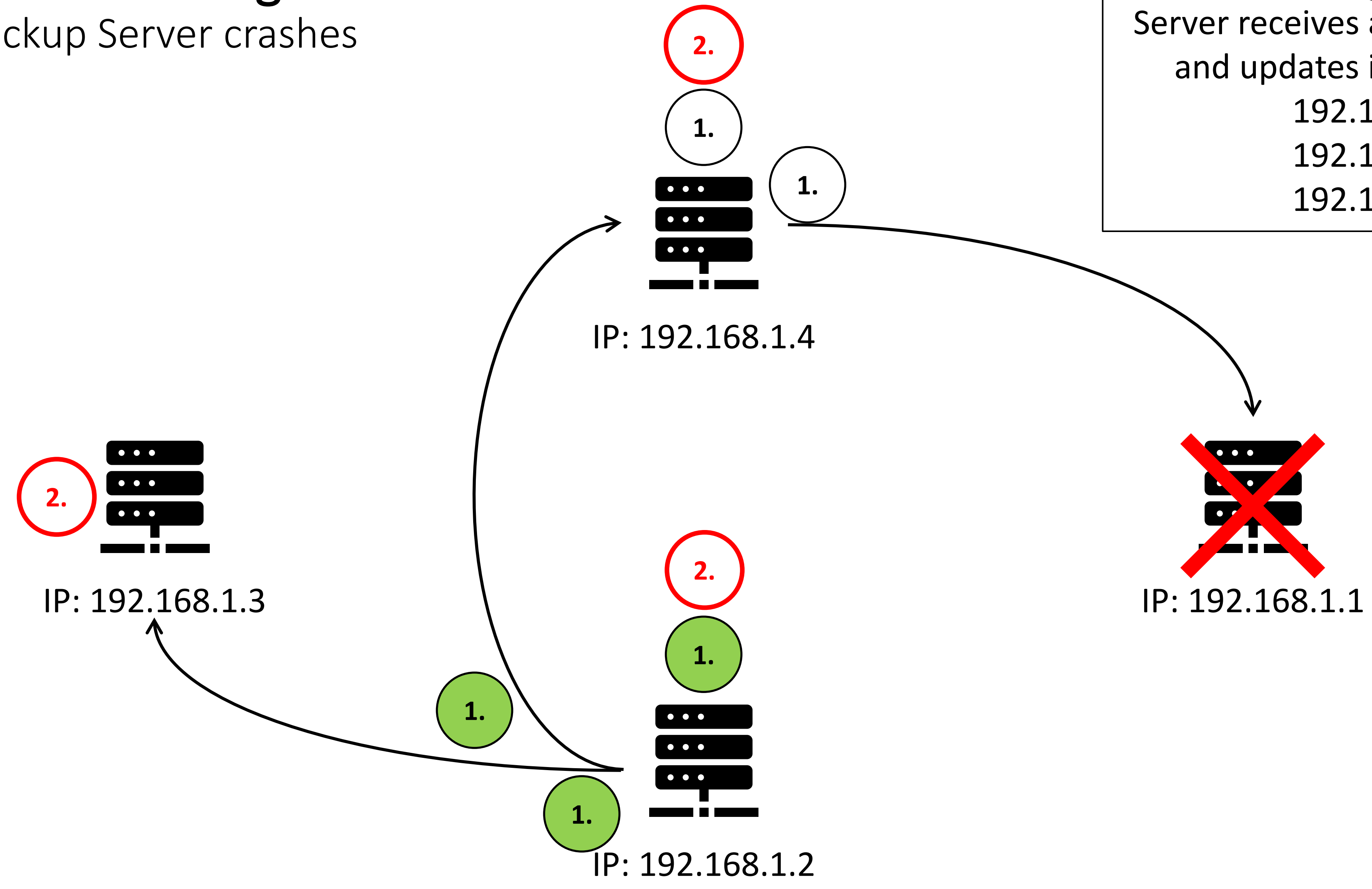


1.

Backup Server notices a timeout threshold of the interval; It broadcasts a failure message with the counter-clockwise neighbour ID FF(CCNid): FF(192.168.1.3)

Heartbeat messages

One Backup Server crashes



2.

Server receives a failure message and updates its group view:

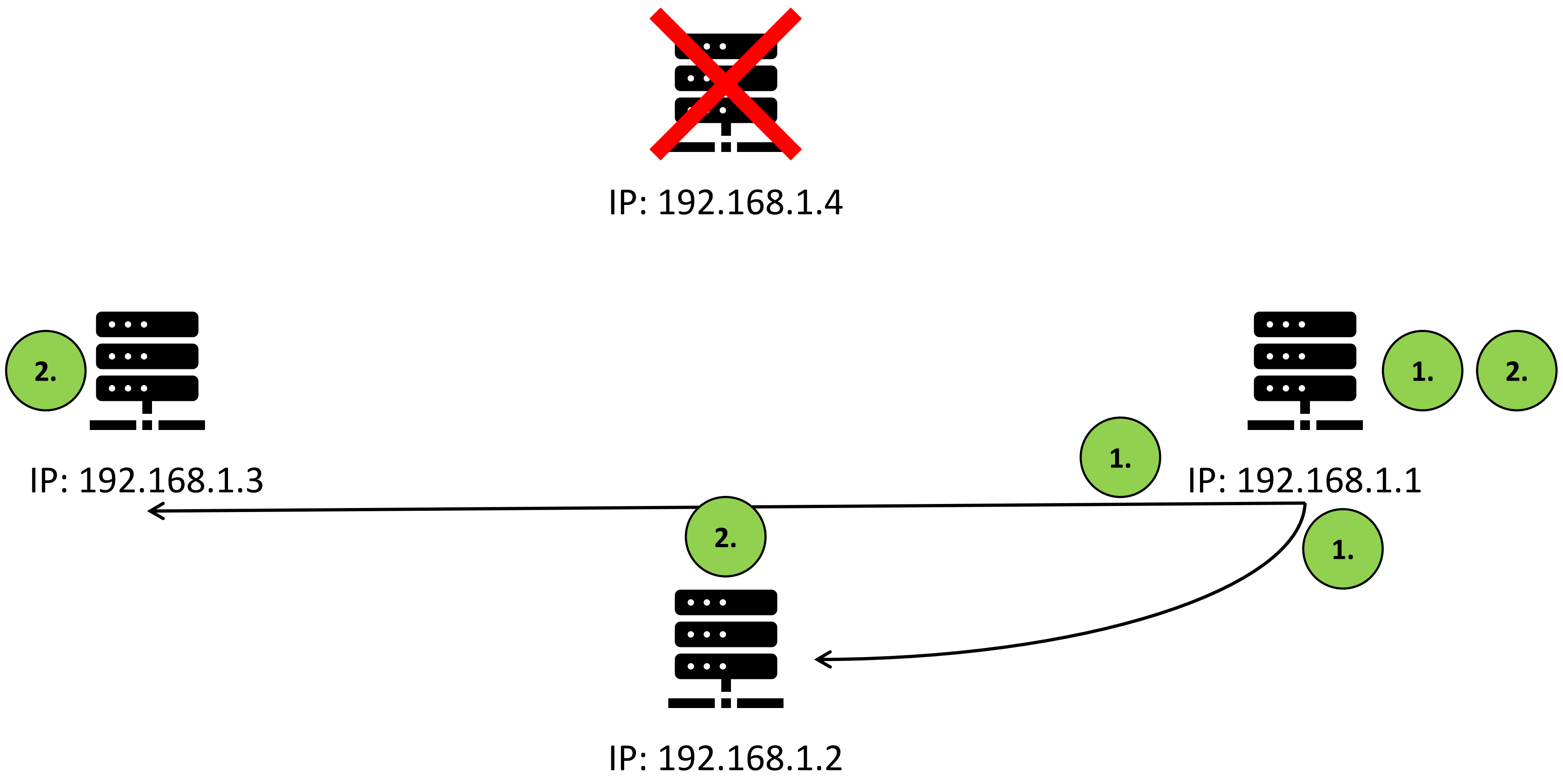
192.168.1.4

192.168.1.2

192.168.1.1

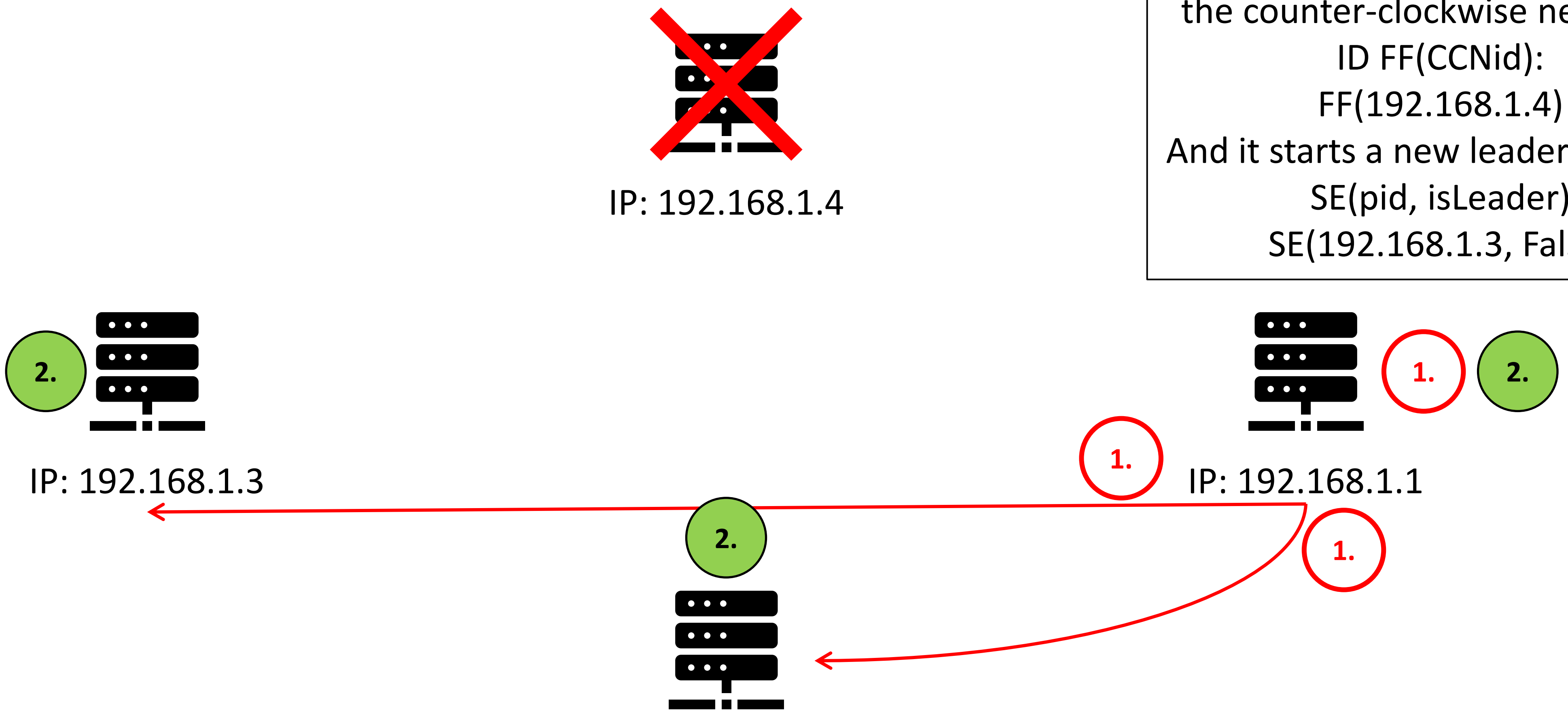
Heartbeat messages

The Primary Server crashes



Heartbeat messages

The Primary Server crashes



1.

Backup Server notices a timeout threshold of the interval; It broadcasts a failure message with the counter-clockwise neighbour ID FF(CCNid):

FF(192.168.1.4)

And it starts a new leader election.

SE(pid, isLeader)

SE(192.168.1.3, False)

Heartbeat messages

The Primary Server crashes

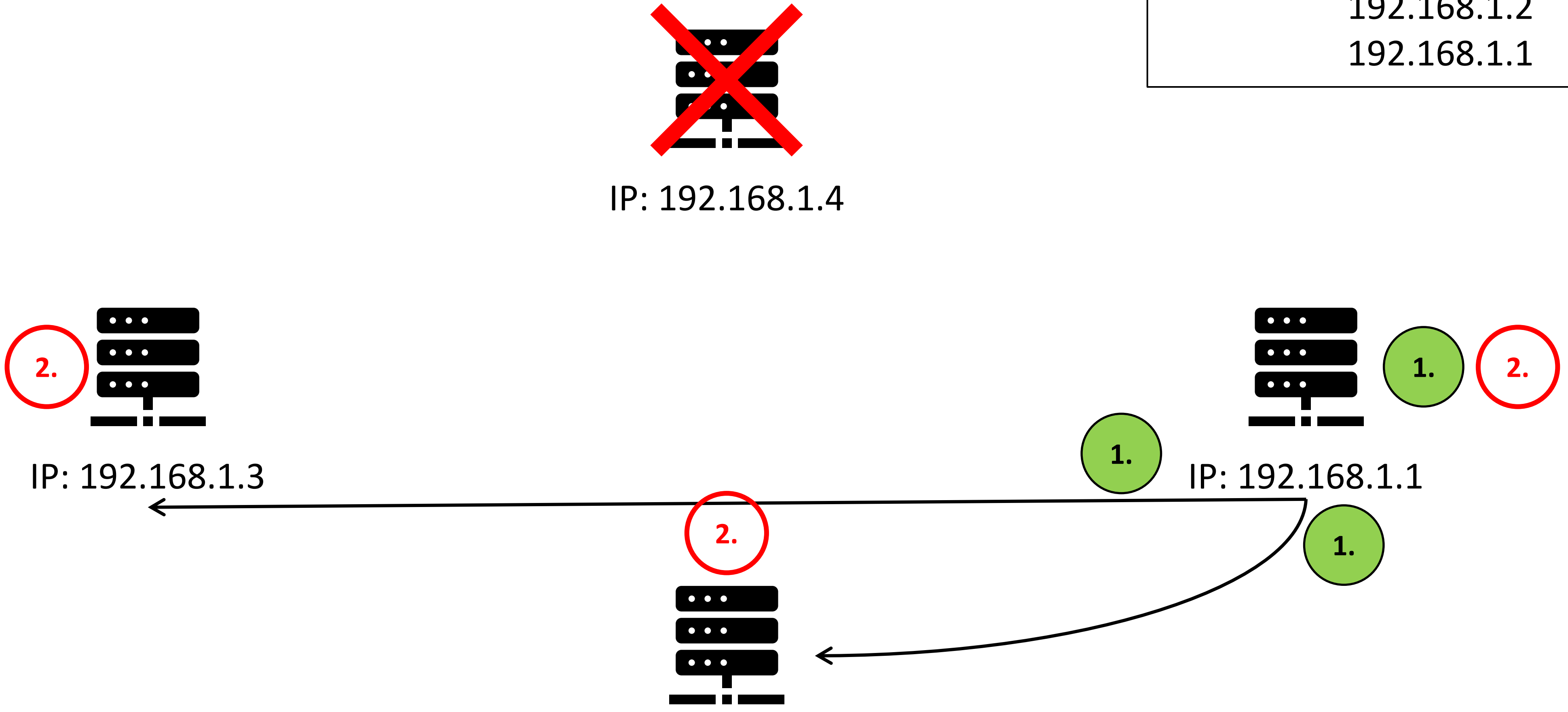
2.

Server receives a failure message and updates its group view:

192.168.1.3

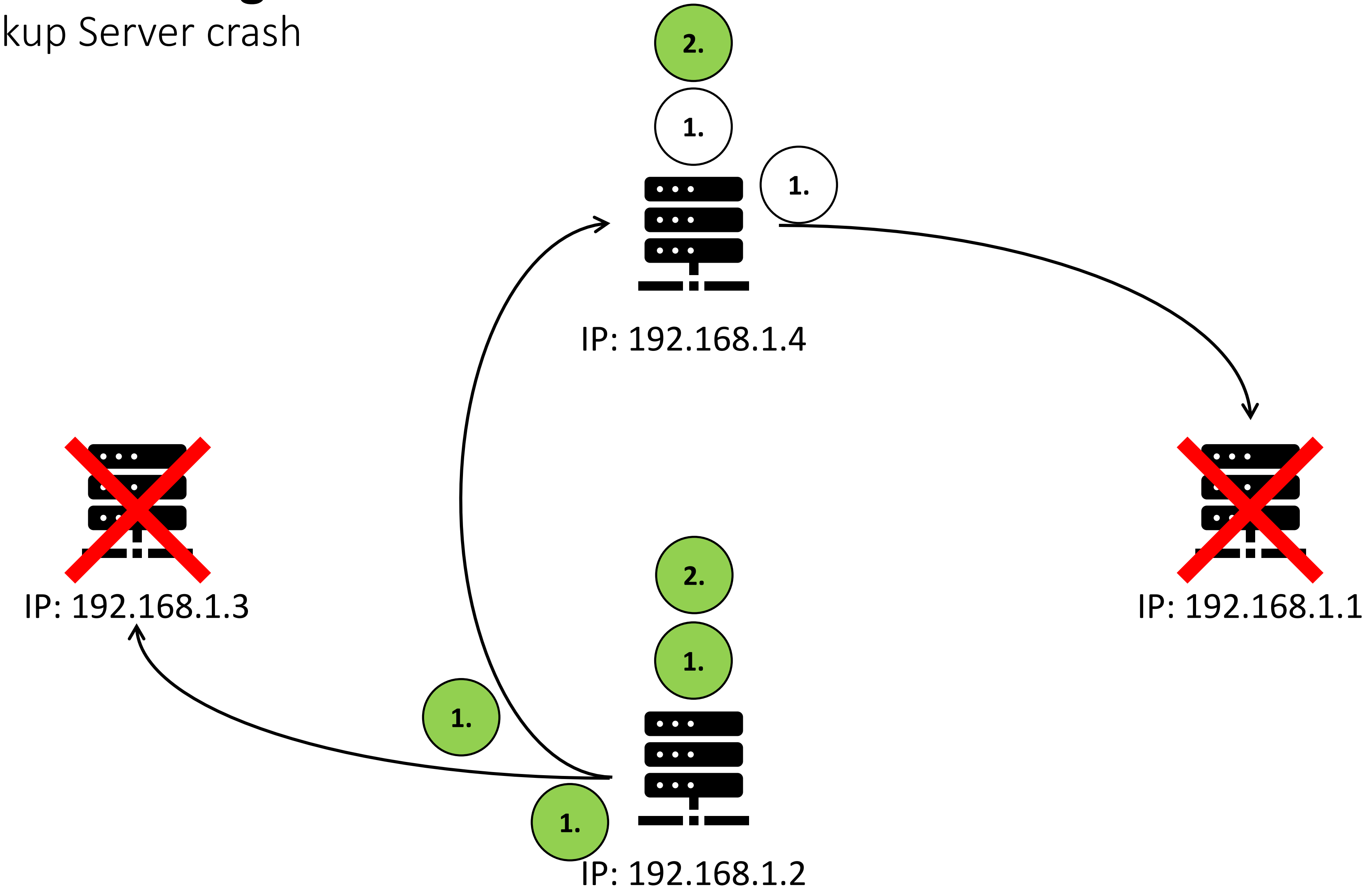
192.168.1.2

192.168.1.1



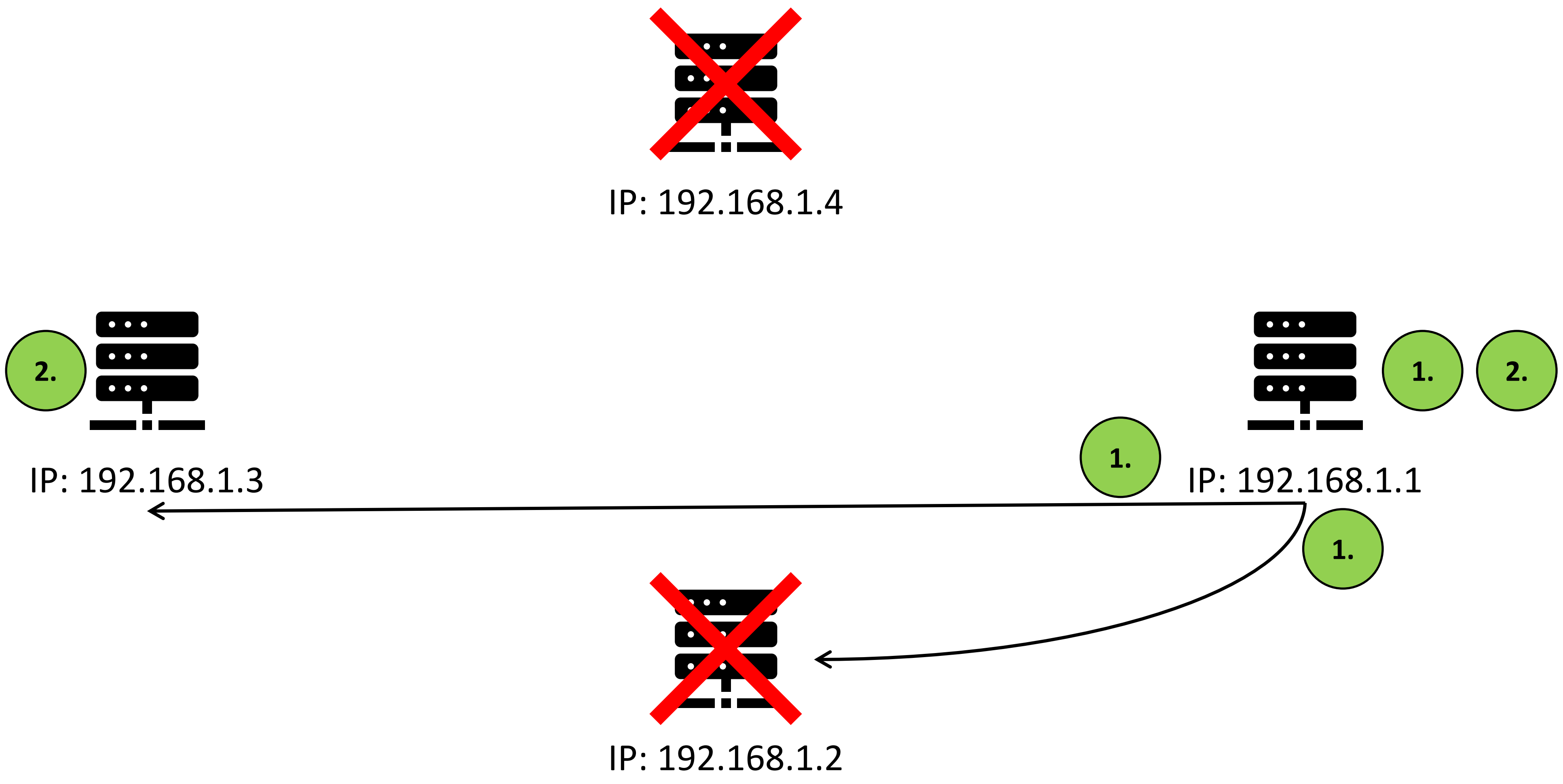
Heartbeat messages

Two Backup Server crash



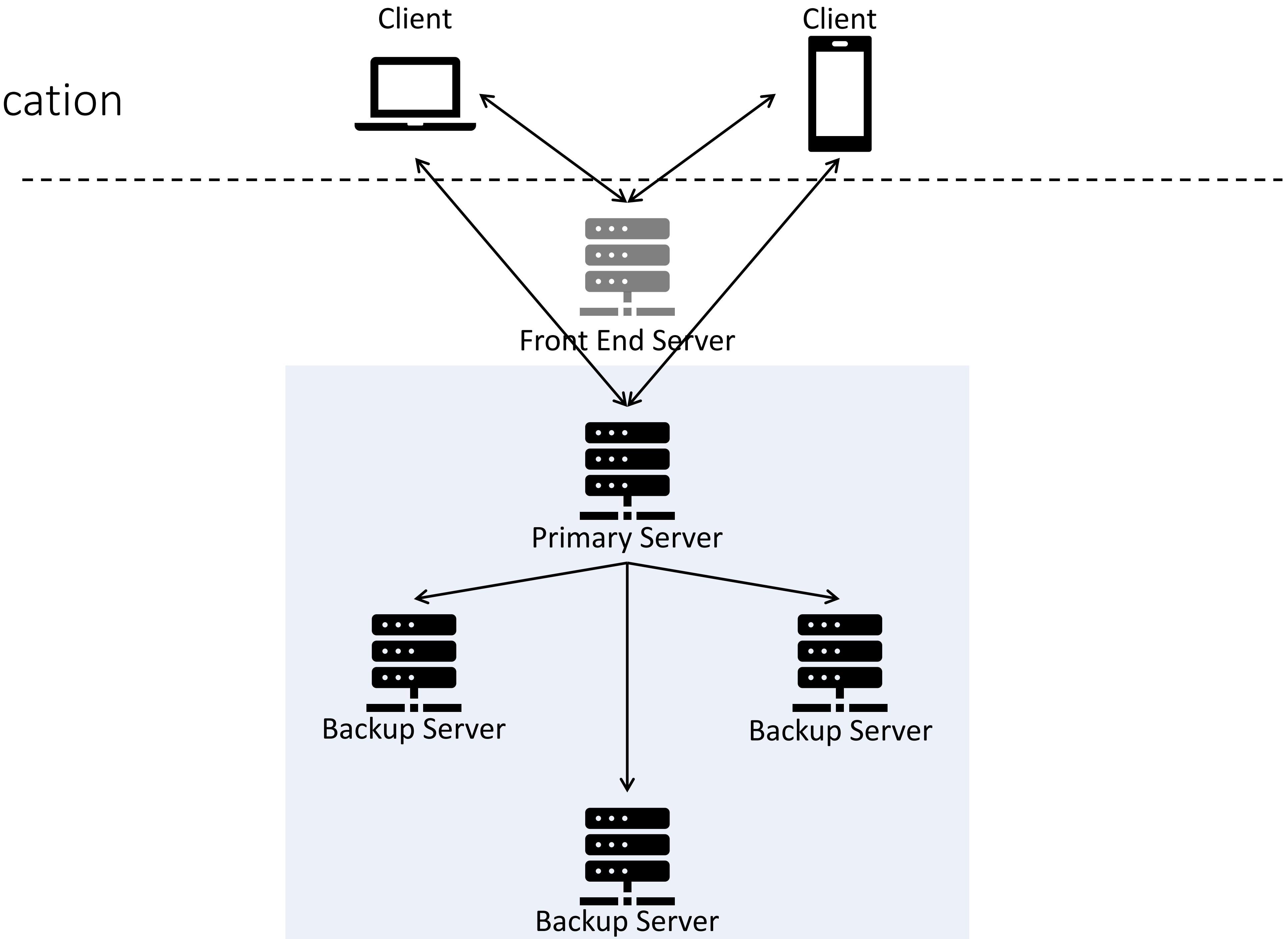
Heartbeat messages

Two Server crash including the Primary

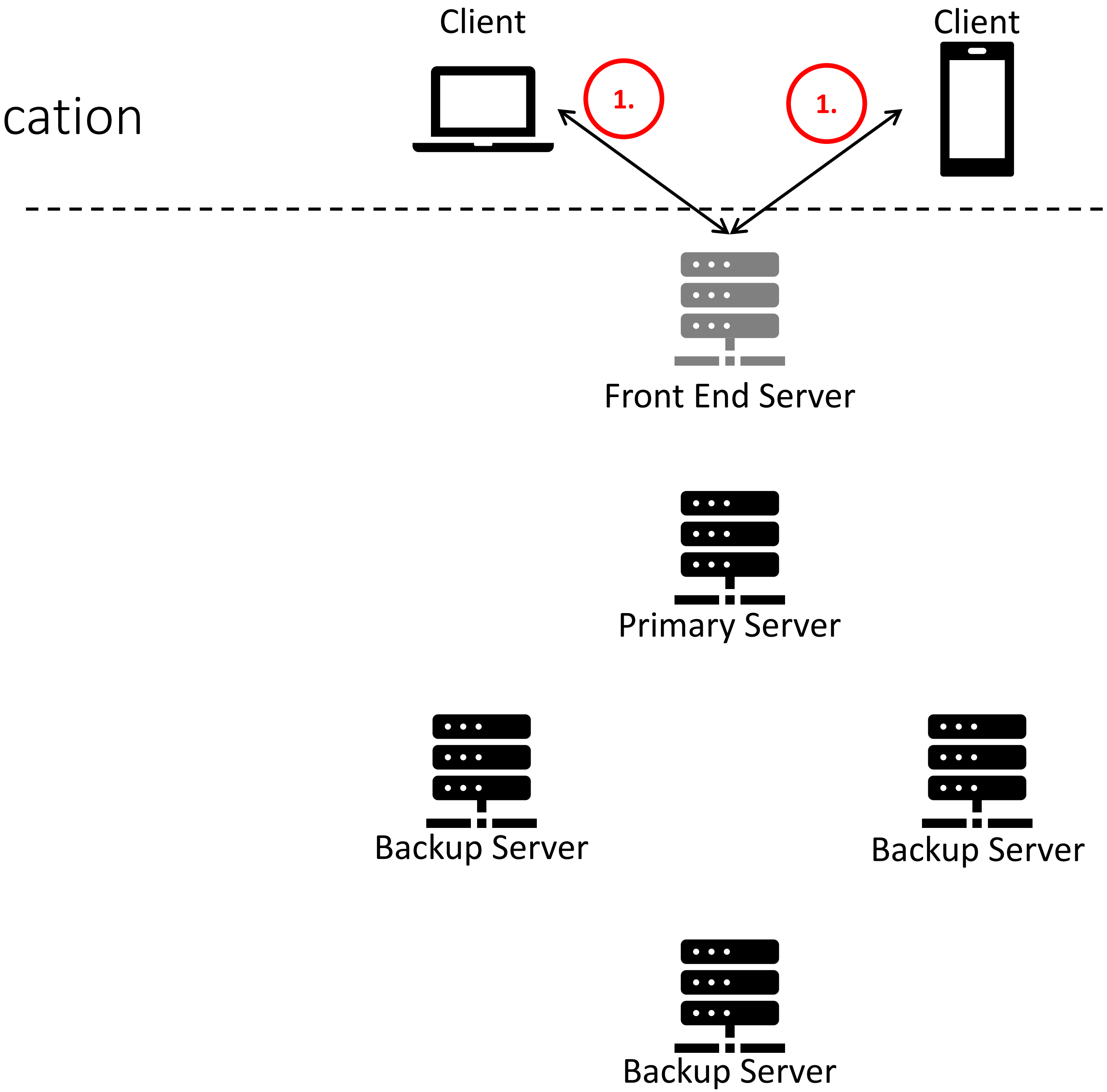


Replication

Replication



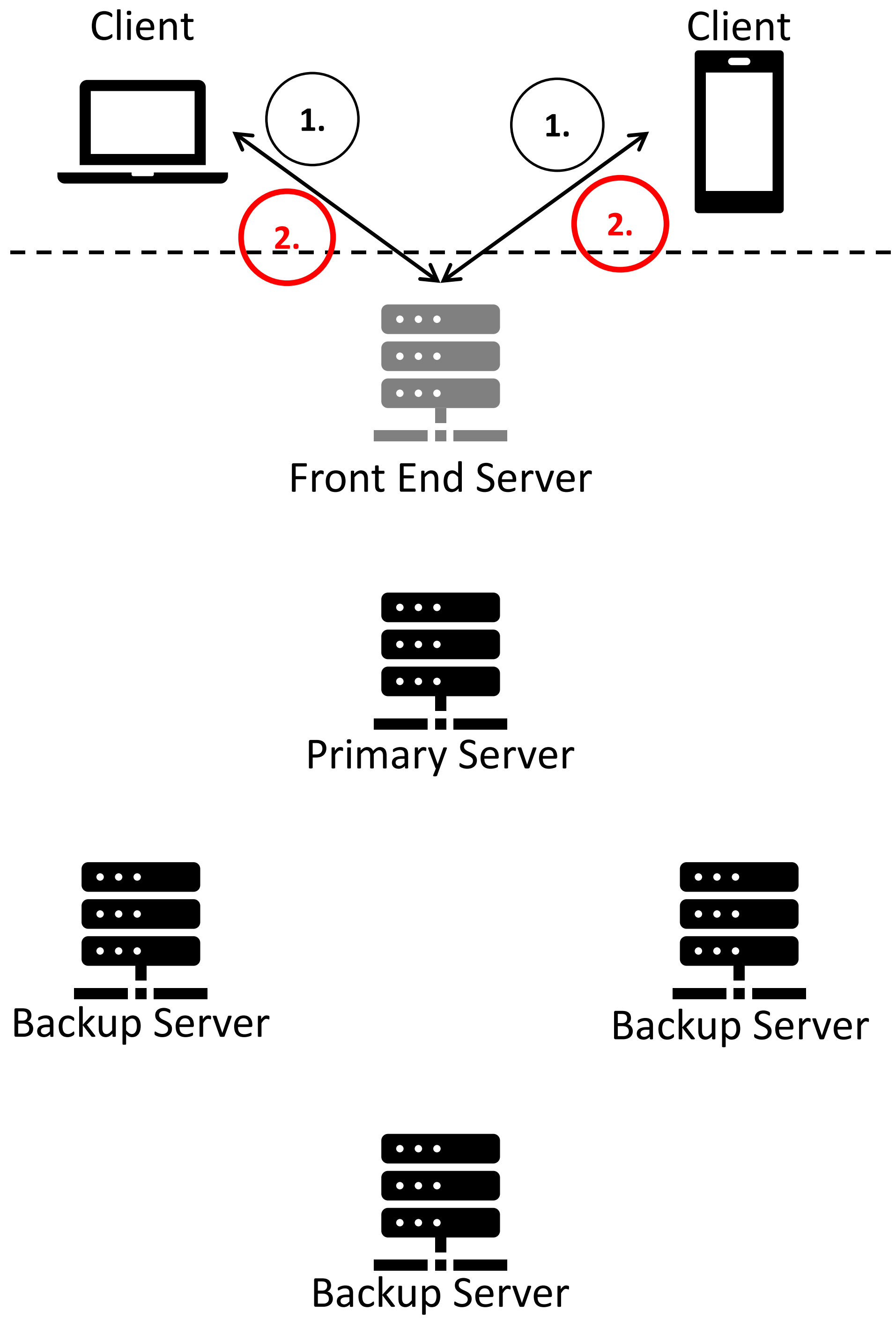
Replication



1.

Clients ask the front end about the primary server.

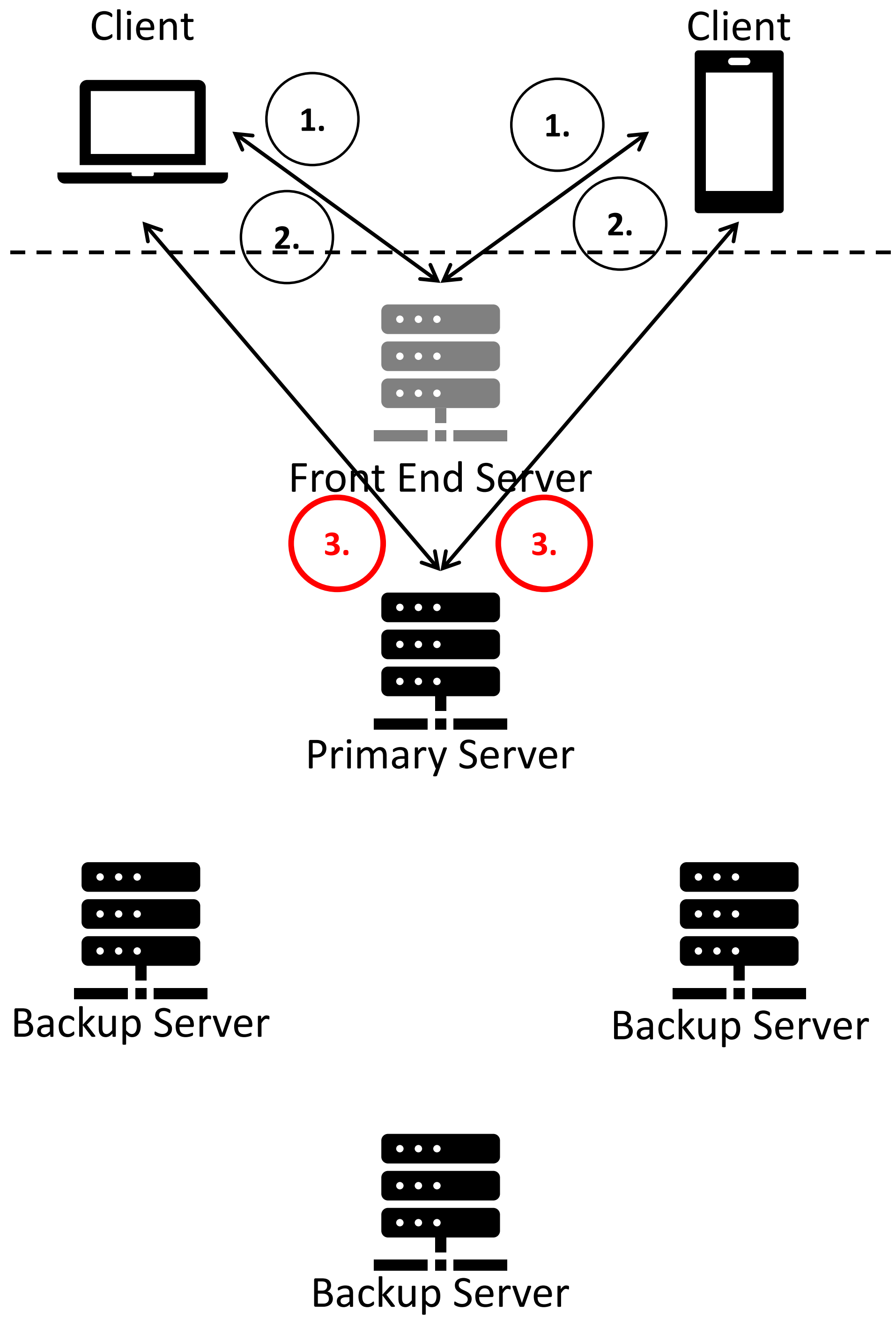
Replication



2.

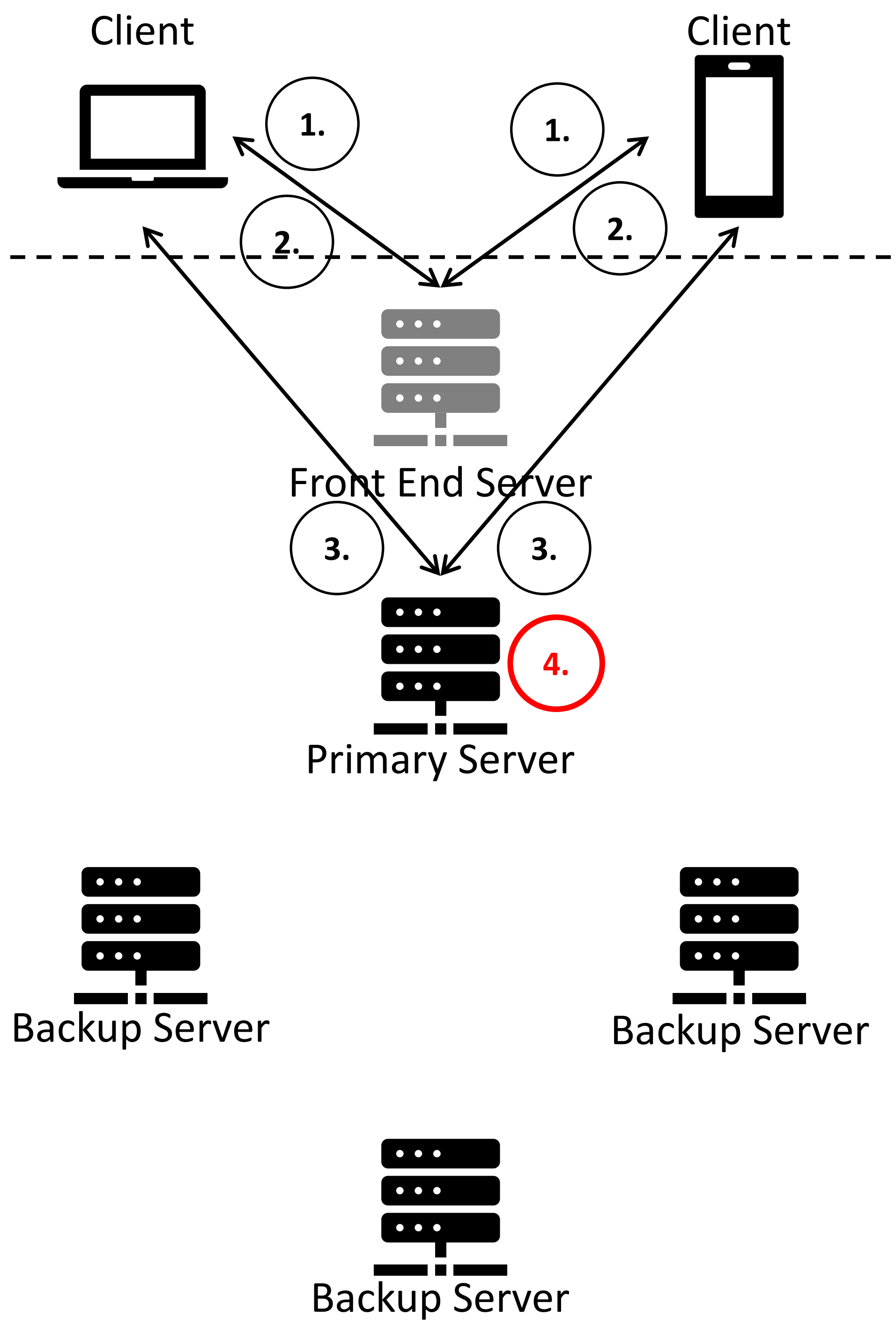
Front end replies the primary servers ID

Replication



3.
The clients share content (CO) to the primary server
CO("content")

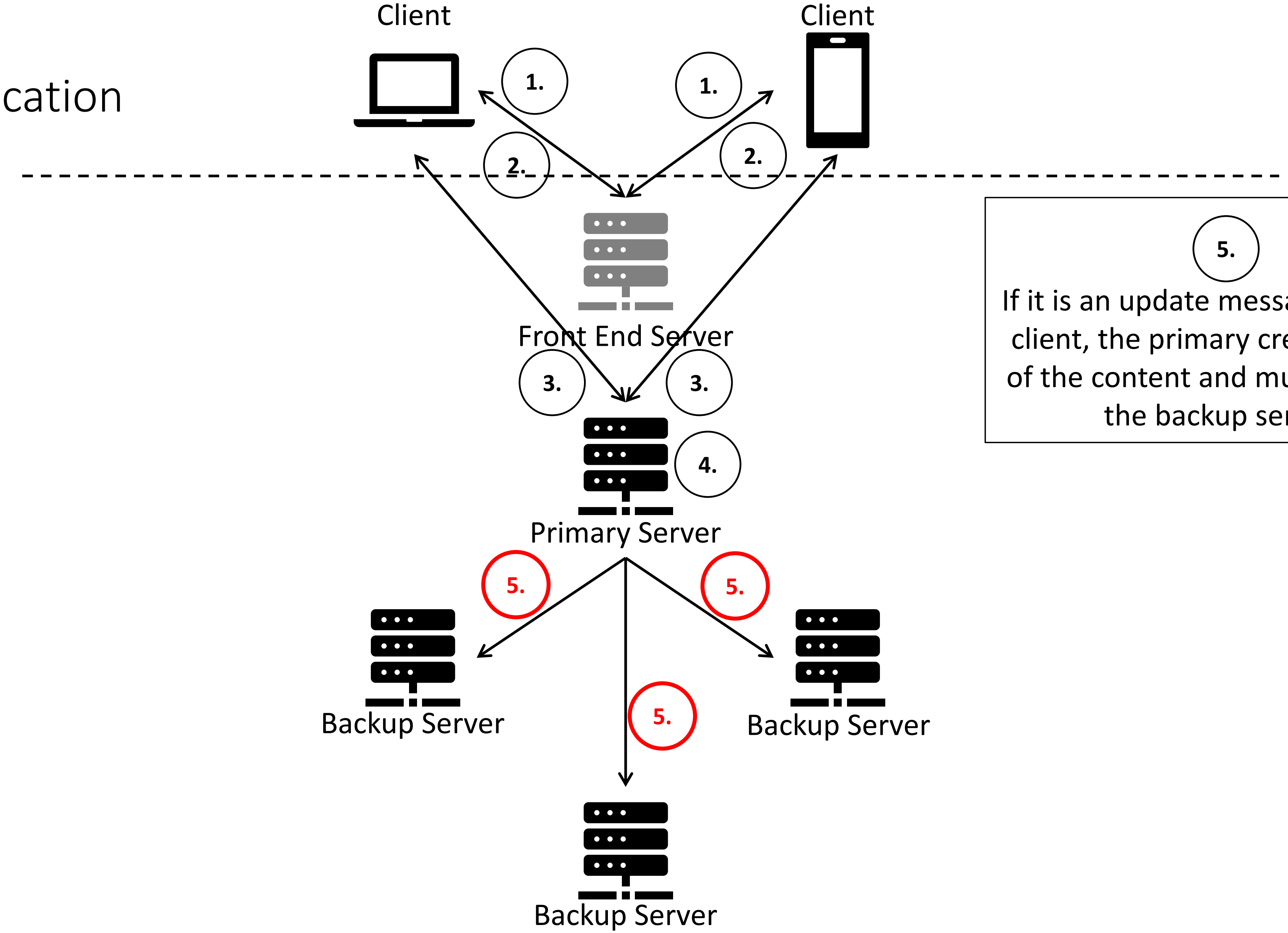
Replication



4.

The primary uses causal ordering to order the incoming messages. He also checks if he already received the message from the client.

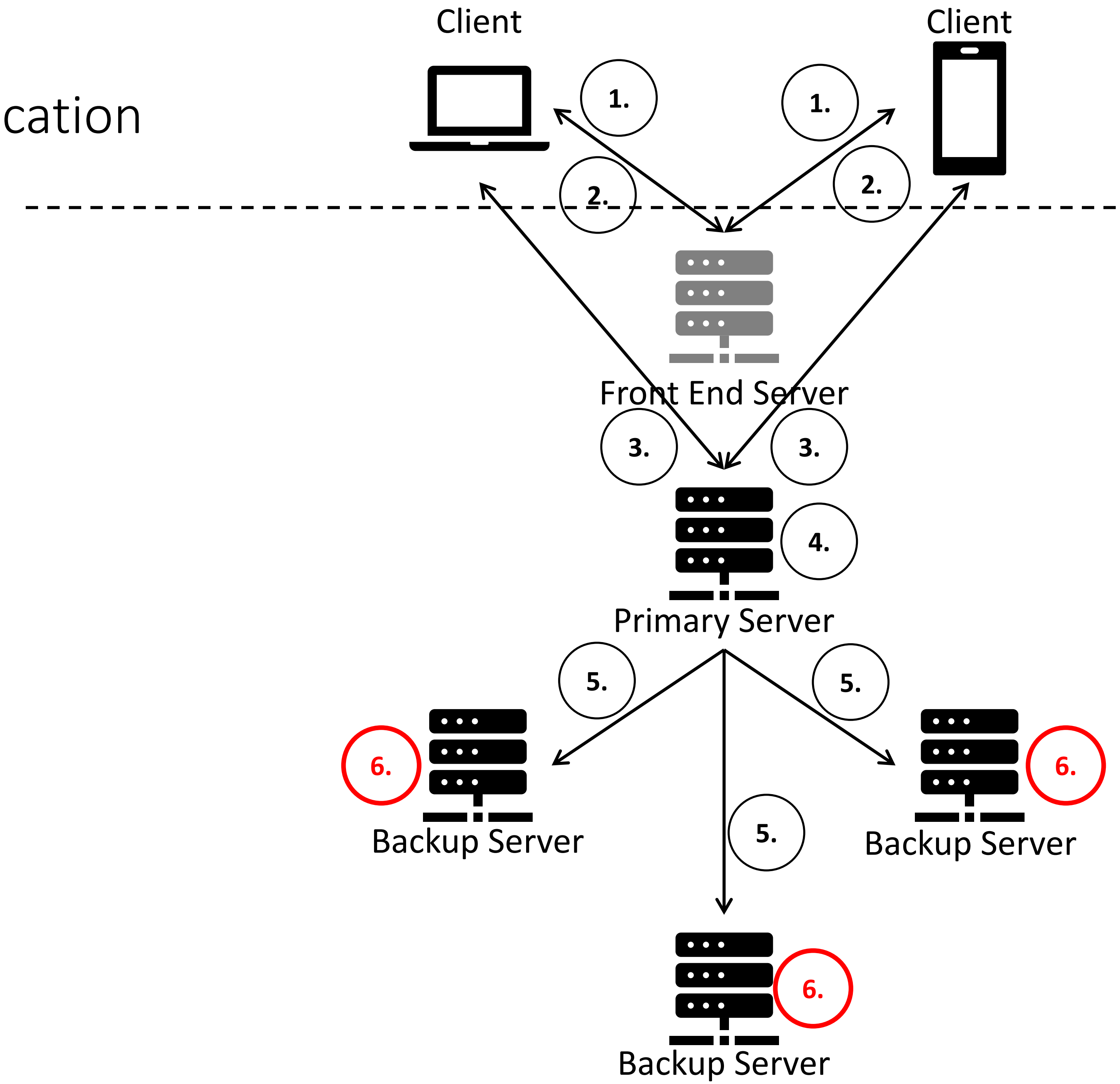
Replication



5.

If it is an update message from the client, the primary creates copies of the content and multicasts it to the backup servers

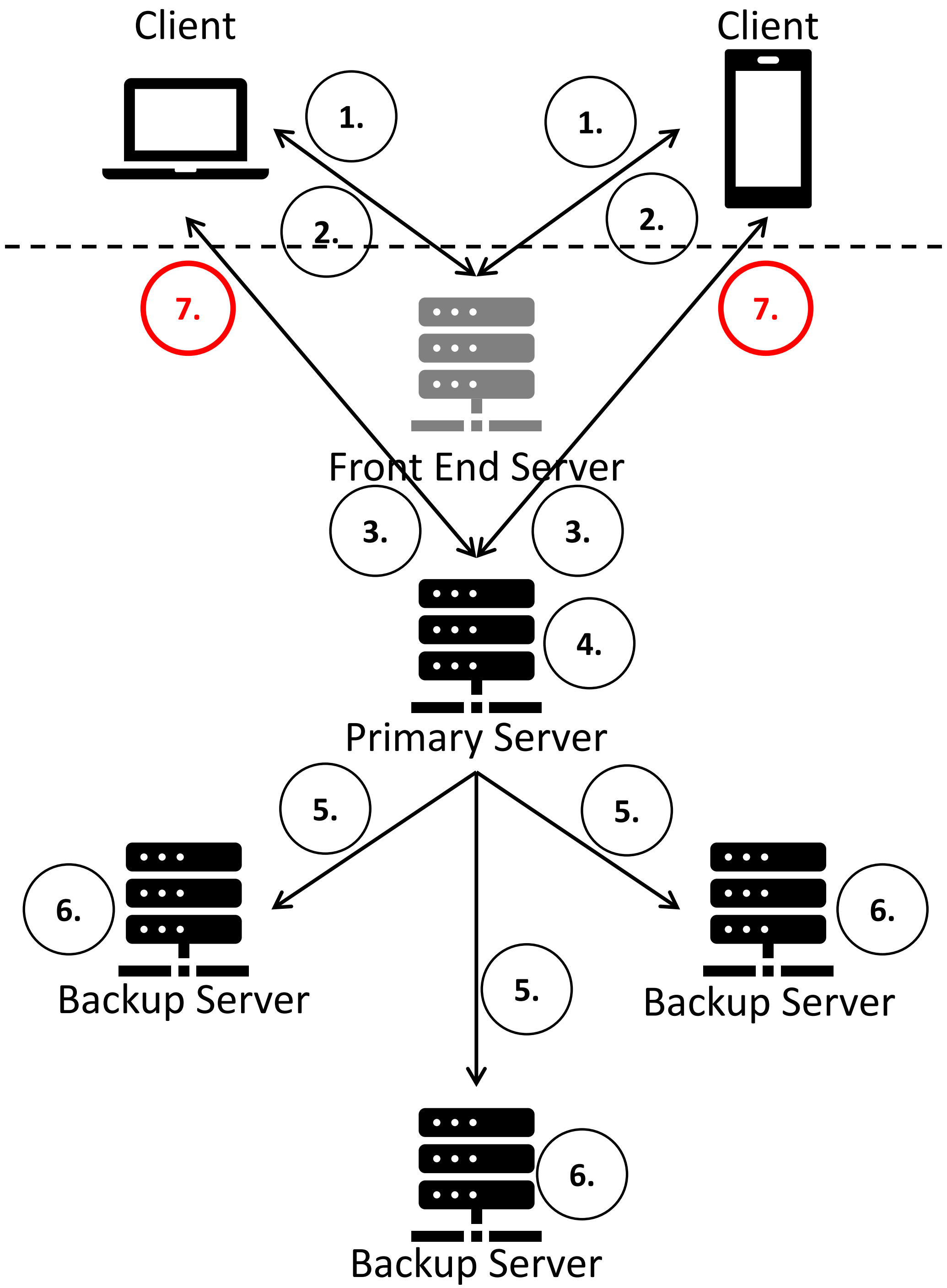
Replication



6\.

The backup server save the updates received from the primary server.

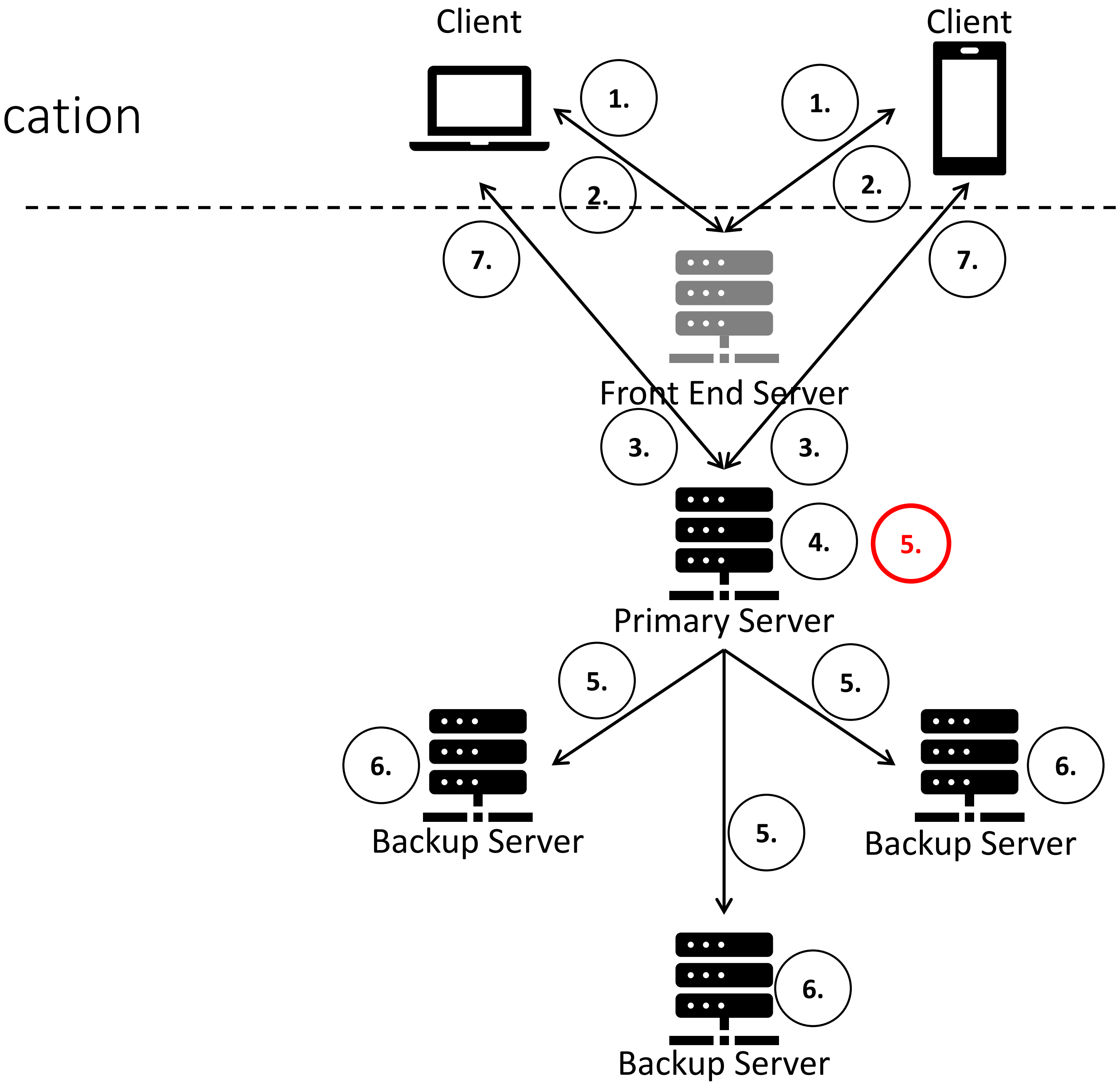
Replication



7.

The primary sends back the updated view to the clients.

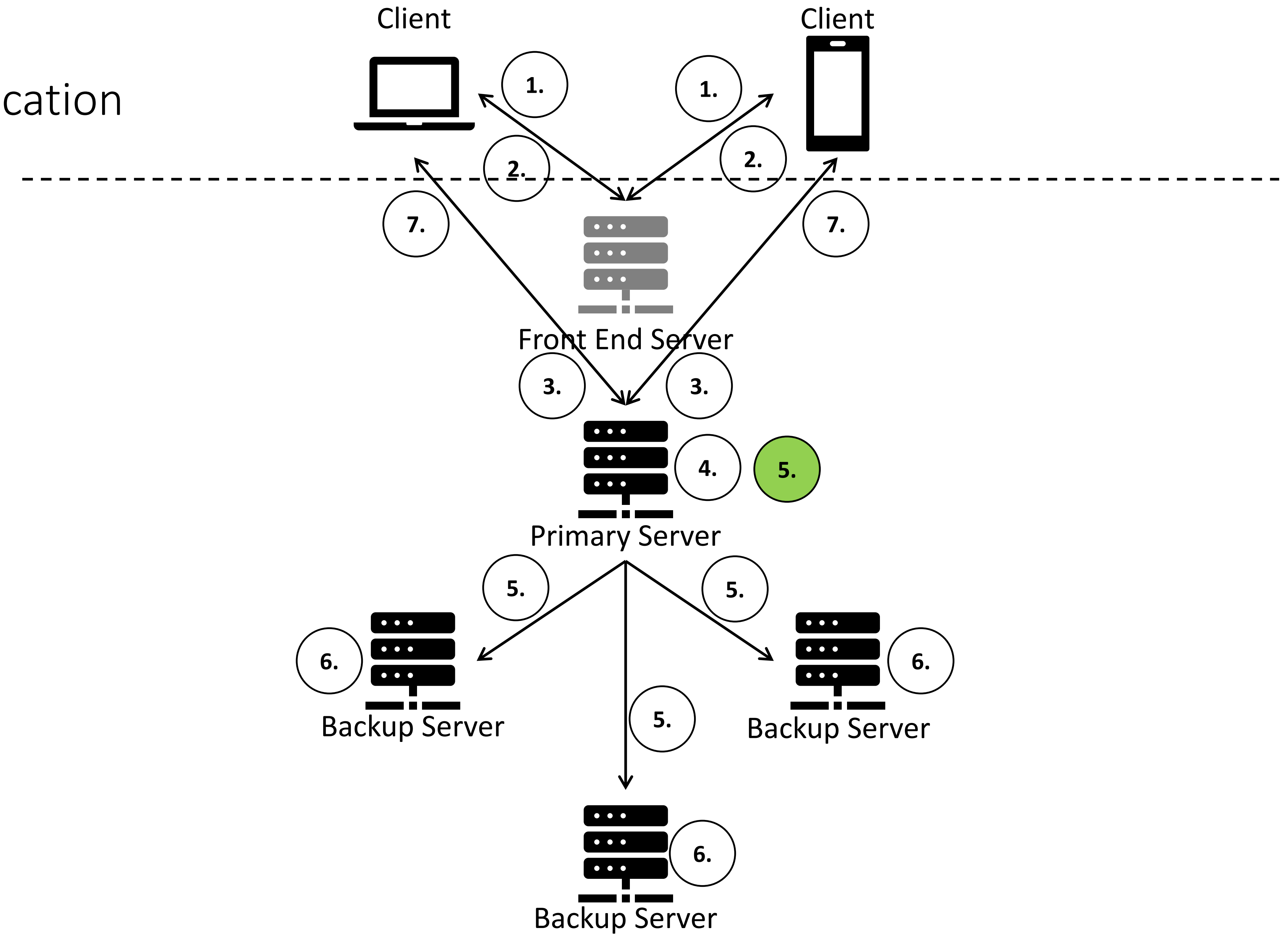
Replication



5.

If the primary already got the received content from the client he does not do anything.

Replication



Client
Client
1.
1.
2.
2.
7.
7.
Front End Server
3.
3.
4.
5.
Primary Server
5.
5.
5.
6.
6.
6.
Backup Server
Backup Server
Backup Server